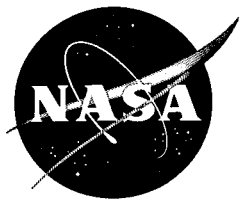


JOHN F. KENNEDY
SPACE CENTER

GE 635
PROGRAMMER'S
REFERENCE
MANUAL

SCIENTIFIC COMPUTATION

SYSTEMS QUALITY BRANCH MANUAL

Communications & Instrumentation

Support Services

Contract NAS 10-4967

Prepared for

National Aeronautics and Space Administration

John F. Kennedy Space Center

by

Federal Electric Corporation

Electronic Data Processing Department

Systems Quality Branch

Revision History

Original - Published 7 January 1972

PREFACE

Although this manual is designed to familiarize the GE-635 user with the configuration and operation of the overall system, the prime objective is to consolidate the most significant information toward work submission, programming standards, restrictions, testing and debugging and related general information as a reference for the GE-635 programmer.

PROGRAMMING REFERENCE MANUAL

INDEX

SECTION 2 PROGRAMMING STANDARDS AND PROCEDURES Page

SECTION 3 TESTING AND DEBUGGING AIDES

SECTION 4 SYSTEMS INFORMATION BULLETINS

SECTION 4 SYSTEMS INFORMATION BULLETINS Page

SECTION 1 GENERAL

1.1 GE-635 COMPUTER SYSTEM

The term "third generation computer" is used frequently to describe the GE-635. This statement is not complete if used in reference to the hardware alone. The "third generation" of data processing is really a generation of multiprogramming operating systems as far as the programmer is concerned.

The applications programmer is able to code a program as though it will be run as a stand alone job. It helps all concerned, however, if you keep one eye on the amount of tape handlers you design into the program and use as few as possible. The operating system (GECOS) will perform the I/O functions required, by working with the GEFRC subroutines that were attached to your slave program when it was loaded. This is the same service programmers have come to expect over the past few years. The main function of GECOS, however, is to keep as many programs as possible in a state of execution at the same time. The tool that GECOS uses to effect total control over all slave programs in memory is interrupt handling.

The interrupts may be caused by:

Timer Runout 12.5 microseconds elapses during the processing of any one slave program.

A change in the status of a peripheral device (operation termination).

Road Block - a request for input causes a temporary halt of processing.

Relinquish - the voluntary release of control by a slave program to GECOS.

Fault - the slave program has violated a GECOS rule (memory protect, privileged instruction, invalid I/O request).

By manipulating these interrupts and the use of special instructions that may only be executed in the master mode, GECOS can determine when, where, and for how long any program will be executed.

1.1 GE-635 COMPUTER SYSTEM (Cont.)

CONSIDERATIONS FOR SYSTEMS DESIGN

The multiprogramming environment of the GE-635 requires a new and different program design approach. Previous systems made no provisions for more than one program to be executed at any one time; therefore, a programmer could assume he had all of core and all peripherals at his disposal.

The philosophy of third generation computers is based on an operating system, which can increase throughput by using peripherals to their maximum capacity. This is achieved by the Allocator, and by limiting the amount of processing in any one program to 12.5 microseconds or until the program is waiting for an I/O function. Therefore, if more programs are in core at the same time, optimum utilization of peripheral devices can be obtained.

ALLOCATION

The allocation of memory and I/O devices is a subject to be carefully considered, because these factors play a significant part in determining how quickly the job will be run after being read into the computer. Some points to consider are:

Be conscious of the amount of core used by your program and insert an accurate core limit in the $ LIMIT GECOS Control Card.

The quantity of tape files requested has a significant bearing on job turn-around time. If at all possible use a disc file instead of a tape file.

PROGRAM SEGMENTATION

Because of the large amount of core and peripheral devices on the GE-635, systems designers may tend to assume that the best approach to program design would be to compact as many functions as possible into one program. This is not true.

Separate jobs may be executed concurrently but separate activities will be executed consecutively. If it is possible to break a program into separate jobs or activities with reduced core and peripheral requirements, your job will be completed much faster.

1.1 GE-635 COMPUTER SYSTEM (Cont.)

PROGRAM SEGMENTATION (Cont.)

A program may be broken into logical segments and use overlay processing via the $ LINK Control Card.

SYSTEM BLOCK DIAGRAM

Figure 1.1 is a block diagram of one GE-635 Computer, Room 205, CIF Bldg., KSC.

This computer complex consists of two GE-635 Computer operating in a dual arrangement.

GE-635 EQUIPMENT SUMMARY

This section contains descriptions of each major component and a general theory of operations.

1.1.1 GE-635 SYSTEM CONTROLLER AND MEMORY

This module contains the magnetic core memory of the GE-635 computer system and provides memory ports for the other principal modules (Processor, IOC and RT-IOC) of the system. The memory module for this system contains 192K words with 36 bits and one parity bit per word. The memory module is capable of performing clear-write and read-restore operations with a cycle time of one microsecond. A total of eight memory ports can be activated, thus allowing for the future introduction of four more principal modules communicating with this memory in addition to the four already in use in the installation.

The memory module also contains the chain of program execute interrupt cells. The cells are set at random by external devices that permit the asynchronous initiation of the execution of many different interrupt handler routines. Each of these routines may pass control to other responding program modules of the user program. The initial memory module of this system contains 16 such interrupt execute cells and an additional 16 cells can be added to support future growth of the system.

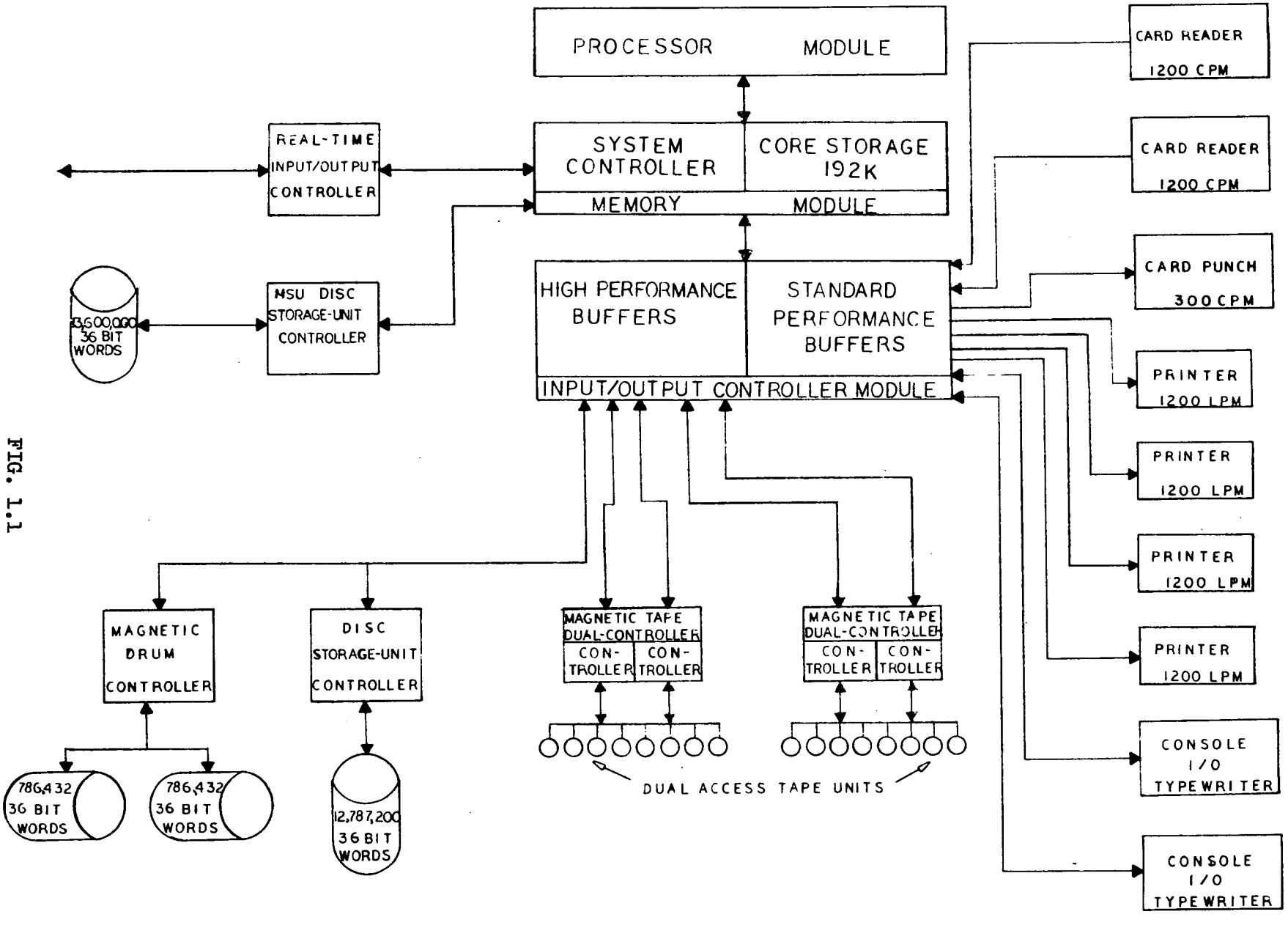


FIG. 1.1

GE 635 COMPUTER SYSTEM

R. BECK, V. FORBES
24 SEPT. 1969

1.1.2 MSU DISC AND CONTROLLER

Mass Storage Unit (MSU) at this complex is comprised of (8) Eight Disc Units and (1) One Controller. Storage Capacity of this MSU is 13.6 Million 36 bit words with an effective transfer rate of 102,400 words/sec.

1.1.3 PROCESSOR

The processor module of the GE-635 computer responds to program execute interrupt cells set and acknowledged in the memory module and carries ou the execution of the program module holding control at a given time. The processor operates in either the master mode or the slave mode, depending upon the executive verxux object program identity of the program holding control, and hence, permits itself to execute all or only a portion of its total instruction repertoire, on the basis of which mode it is in at a given time. The processor not only can access all of the initial 192K memory but as new memory modules are added to the system, it can directly access up to a total of 262,144 words of magnetic core memory distributed among four memory modules.

1.1.4 INPUT/OUTPUT CONTROLLER (IOC)

This module provides the interface for the common peripheral sub-systems used with the GE-635 computer. It can be expanded to permit up to four memory modules to communicate with up to sixteen peripheral device controllers. There are 6 maximum speed channels rated at 400,000 character/second and 10 standard channels rated at 250,000 character/second. The IOC module of this system initially contains implemented I/O channels for each of the common peripherals enumerated in Figure 1.

1.1.5 REAL-TIME INPUT/OUTPUT CONTROLLER (RT-IOC)

This module provides the interface between the GE-635 computer and the external real-time devices of the system in which the computer is used. It can be expanded to communicate with up to four memory modules and hence access up to 262,144 words of magnetic core memory. The RT-IOC permits a number of external devices to share the data transfer

1.1.5 REAL-TIME INPUT/OUTPUT CONTROLLER (RT-IOC) (Cont.)

capability of a memory port and therefore the number of I/O Channels implementable in a given RT-IOC application is set by the aggregate data rate of the external devices being considered. The initial RT-IOC of this system can capture every second memory cycle to obtain control words and for the purpose of transferring up to 36 bits of data per memory access. However, the real-time devices of this installation (notably the Data Block Address channels of the Data Core Adapter) can experience peak load conditions which require a significant portion of this capability and hence the initial RT-IOC of this system has been limited to a total of twelve I/O channels which may be classified as follows: six payload channels in use, four overhead channels, and two spare payload channels available for additional low data rate devices.

The initial KSC RT-IOC is a special design in the sense that it can capture every second GE-635 memory cycle, while the normal RT-IOC design can only capture every third GE-635 memory cycle. In exchange for this higher data rate, the following system design constraints have been observed:

No more than twelve I/O interrupt cells can be sampled, and hence limits such as RT-IOC to a maximum of twelve I/O channels.

No double cycle transactions can be performed. This is to guarantee capture of the RT-IOC in time to service the DBA channels of the Datacore adapter. Furthermore, the indirecting option cannot be introduced to this particular RT-IOC since indirect demands two consecutive RT-IOC cycles and could thus prevent capture of the RT-IOC in time to service a DBA. Present channels which require a store-execute capability are required to capture the RT-IOC twice, thus opening the way for capture by the higher priority DBA channels. The RT-IOC is supported by a control block of 512 locations assigned to the master mode portion of memory. The The initial KSC RT-IOC can set eight different program execute interrupt cells in memory, as defined in memo CIF-PM2.2.3.

1.1.6 DATACORE ADAPTER

The datacore adapter is an external device served by three of the payload I/O channels of the RT-IOC and is used to interface the customer-furnished Datacore system with the computer system. This function consists of advicing the real-time program as to which telemetry blocks have filled in Datacore and are now ready to be "dumped" into the computer and secondly to carry out this transfer to telemetry data to the computer.

Since Data Block Addresses (DBA's) can occur in a burst of one every 4 microseconds for short periods, the datacore adapter is designed to receive the DBA's and then enter the essential information into a 64 word flag table in memory via one of two I/O channels used only for this purpose. The first channel is called the "basic" channel and will process all DBA's when they occur less often than on consecutive output cycles of the Datacore. The second channel is called the "relief" channel and will process every second DBA when they do occur at the maximum rate of one every 4 microseconds. These channels enter a flag bit into a specific bit position of a specific flag word to indicate which one of 64 possible telemetry block addresses has occurred and a second bit into another position of the same word to indicate if that telemetry block has filled while in the upper or lower buffer to Datacore.

The third I/O channel serving the datacore adapter is used to convey Data Control Words (DCW's), which have been fabricated by the real-time program, to the adapter so that it may interrogate Datacore and "count in" the corresponding number 12-bit telemetry words residing in each of the filled blocks of Datacore. The input half of this same channel is used to store the telemetry data after the datacore adapter has packed them, three to a computer word.

The datacore adapter is designed to request the RT-IOC to set a program execute interrupt cell in the GE-635 memory module whenever the datacore adapter has completed storing of all the telemetry data blocks detected on a given scan of the flag table.

The datacore adapter continously displays a "busy/notbusy" status bit which the RT-IOC can sample when instructed by the real-time program. This bit will be in the busy state while the datacore adapter is transferring all of the telemetry blocks detected on a given scan of the flag table. (See Fig. 1.2)

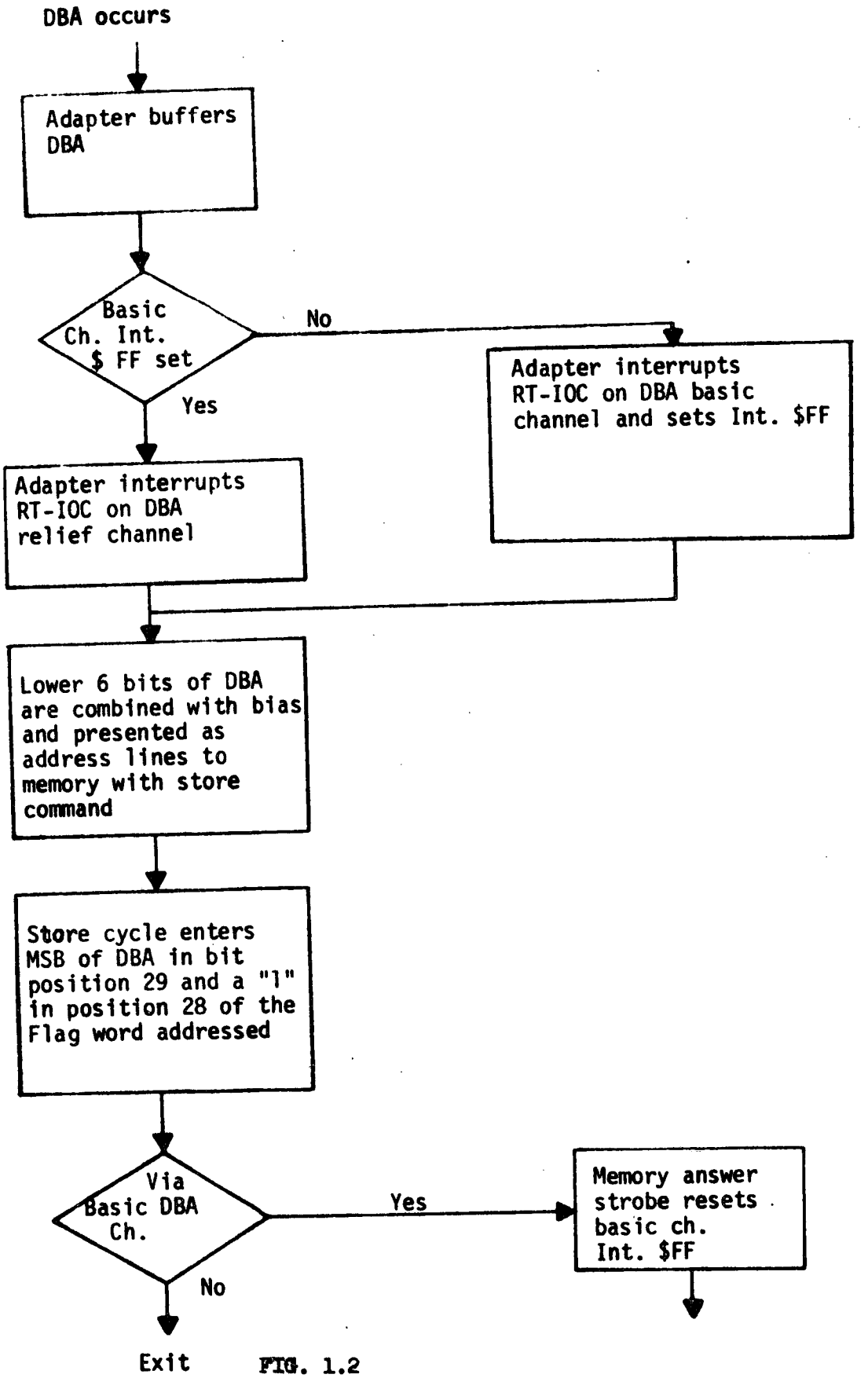

DBA occurs
Adapter buffers DBA
Basic Ch. Int. $ FF set
No
Yes
Adapter interrupts RT-IOC on DBA basic channel and sets Int. $FF
Adapter interrupts RT-IOC on DBA relief channel
Lower 6 bits of DBA are combined with bias and presented as address lines to memory with store command
Store cycle enters MSB of DBA in bit position 29 and a "1" in position 28 of the Flag word addressed
Via Basic DBA Ch.
Yes
No
Memory answer strobe resets basic ch. Int. $FF
Exit

FIG. 1.2

1.1.7 MAGNETIC TAPE CONTROLLERS

Two controllers possessing two channels each are required Each controller can serve sixteen tape transports and hence are referred to as 2 X 16 or cross-barred controllers.

1.1.8 MAGNETIC TAPE TRANSPORTS

Sixteen tape transports are supplied for use with the above controllers. The transports operate at a 120 KS character rate and are compatible with the IBM 729 Mod VI tape format.

1.1.9 STANDARD DRUM

The two magnetic drums are used to support the standard software of the system. The storage capacity of each drum is 786,432 36 bit words and its rotation speed is 1800 RPM.

1.1.10 PRINTERS

4 on line printers, 1200 lines per minute each.

1.1.11 CARD READER

2 1200 CPM card readers.

1.1.12 CARD PUNCH

1 300 CPM card punch.

1.1.13 OPERATORS CONSOLE

1 Master console and 1 auxiliary console both with I/O typewriters.

1.1.14 DISC

5 discs are on each system. Total Disc Storage of one system is: 12,787,000 36 Bit Words. 3333 Links. 39996 Llinks.

* 12 LLinks = 1 link
1 Link = 3840 36 Bit Words
1 LLink = 320 36 Bit Words

1.1.15 PAPER TAPE PUNCH AND READER

A paper tape punch and reader is provided on one system only.

1.1.16 TIME-SHARING SYSTEM (TSS)

A time-sharing system enables a large number of individuals to make use of an Information System simultaneously. The time-sharing users work with the computer through remote terminals. The users submit programs, control their compilation and execution, and upon request, supply data input through these terminals.

The user "controls" the time-sharing system by means of a command language distinct from any of the specialized programming languages that are recognized by the individual time-sharing compilers and processors. (e.g., the Time-Sharing FORTRAN language). The command language is generally the same for any component of the time-sharing system. (FORTRAN, BASIC, TEXT EDITOR, etc.) Some commands pertain to only one or another of the component time-sharing systems, but the majority are, in form and meaning, common to all component systems.

The primary functions of the time-sharing command language are as follows:

Initiation of processing within a sub-system.

Example - LIST and RUN commands.

Storage, retrieval and purge of permanent files.

Example - SAVE and OLD commands.

Request for operations on temporary time-sharing files.

Example - NEW and RESEQUENCE commands.

Request for pertinent operating information.

Example - HELP and STATUS commands.

Direction of flow of control within sub-systems.

Example - DONE and BYE commands.

This command language is described fully in the following manuals.

TIME-SHARING FORTRAN
TIME-SHARING BASIC
TIME-SHARING GENERAL INFORMATION MANUAL
TIME-SHARING TERMINAL/BATCH INTERFACE

1.1.16 TIME-SHARING SYSTEM (TSS) (Cont.)

The GE-600 Line Time-Sharing System operates under the direction of GECOS III (GE-600 Line Comprehensive Operating Supervisor) and is one demension of an integrated, 3-dimensional information system. Under GECOS III, these three dimensions, Batch, Remote-Batch and Time-Sharing, carry out their activities simultaneously, with intercommunication between the three dimensions. The time-sharing terminal user should note the significance of this intercommunication.

The Time-Sharing System (TSS) is comprised of a Time-Sharing Executive and a number of independent processing sub-systems that operate under the Executive and a common command language. These major sub-systems include the following:

BASIC -- an algebraic-language compiler/executor, designed for the user with numerical problems involving relatively small amounts of data. (See manual CPB-1510)

TSS FORTRAN -- an algebraic-language compiler/loader with extended capabilities for subprogramming, chain overlays, and peripheral I/O, providing full batch-type programming capabilities. (See manual CPB-1566)

Text EDITOR and RUNOFF -- a facility for building, maintaining, and reformatting text files of any description. (See manual CPB-1515)

ABACUS -- a "desk-calculator" facility featuring sophisticated algebraic capabilities such as functions, summation operator, and remembered variables. (See manual CPB-1643)

ACCESS -- a file system manipulation subsystem that allows the user to create, delete, and modify file system cataloga, subcatalogs, and named files. The file space, not file content, is manipulated with ACCESS. (See manual CPB-1643)

RBUG -- a conversational debug routine that can be used in conjunction with CARDIN. RBUG has all of the capabilities of the DEBUG routine of the batch world, permitting the user to monitor execution of his program, insert and remove

1.1.16 TIME-SHARING SYSTEM (TSS) (Cont.)

breakpoints, and alter contents of memory locations and registers dynamically -- all in an interactive manner. (See manual CPB-1646)

SCAN -- provides a means of examining output of a batch job from a time-sharing terminal; the batch job may have been submitted through CARDIN, remote-batch, or as standard-central-site job with its output placed into the file system. (See manual CPB-1642)

CARDIN -- a facility for submitting of a batch job at a time-sharing terminal for batch-world processing; job status information is available on demand. The SCAN subsystem complements CARDIN with its facilities that scan job output. (See manual CPB-1642)

FDUMP -- a remote-terminal, word-oriented file inspection and maintenance facility for permanent files, regardless of their format. The files may have been generated in either batch, remote-batch, or time-sharing environments. (See manual CPB-1642)

FORTRAN TRANSLATOR -- permits the user to translate a time-sharing FORTRAN file into batch FORTRAN. The user may design and debug a program in time-sharing FORTRAN and then optimize its processing by converting it to batch FORTRAN. (See manual CPB-1643)

HELP -- supplies the user, at his terminal, with a detailed explanation of any system error message. (See manual CPB-1643)

Library Editor (LIBED) -- is specifically intended for editing of time-sharing FORTRAN subroutine library files, such files to be subsequently processed by the Library Generator (TSLG) program. (See manual CPB-1643)

Library Generator (TSLG) -- permits a user to produce his own library file of time-sharing FORTRAN subroutines, complete with directory, in a form that is acceptable to the time-sharing FORTRAN loader. (See manual CPB-1643)

1.1.16 TIME-SHARING SYSTEM (TSS) (Cont.)

Media Conversion Program -- is a batch-world program that may be run either at the central computer site or through a remote/batch terminal. Its purpose is to generate a standard format, time-sharing text file from a suitable card deck, or conversly, to produce a card deck from such a file. (See manual CPB-1643)

SECTION 1 GENERAL

1.2 AUXILLIARY EQUIPMENT

This installation uses the UNIVAC 1005 for EAM support as well as the usual card sorting, reproducing and collating equipment. This auxilliary equipment may be utilized by the 635 user for tape dumps, card to tape, card listing, file maintenance, tape copy, etc.

In addition to the above there are card and tape transmission systems available for data transmission to other sites using compatible equipment.

This installation also contains a Stromberg-Carlson 4020 Plotter for plotting GE-635 output. The applications being the rapid production of labeled graphs and its use as a high-speed printer with all information stored on 35 mm film.

See Section 1.3 for forms necessary to request auxiliary support.

See Section 1.4 for manuals regarding this equipment.

1.3 SUBMISSION REQUESTS

1.3.1 COMPUTER WORK REQUEST AND AUXILLIARY EQUIPMENT WORK REQUESTS

Work will be submitted in room 209 of the CIF Building, unless programmer's department has special arrangements elsewhere. Room 209 is known as the I/O counter and both Card Processing Requests and Computer Operations Work Requests are handled through this counter. Blank work requests are also available at the I/O counter.

Key Punch and other Unit Record Requests

To submit data for keypunch refer to Figure 1.3. Figure 1.3 is a Card Processing Requests form. Each request must consist of three copies (1 original and 2 duplicates). The third copy is returned to the user at submittal time after the request is marked received. The second copy is also the user's and is returned to him upon acceptance of his data. The original is kept for job accounting records. Acceptance of the user's data is indicated by signing the user's name or initialling in the appropriate area at the bottom of the Card Processing Request.

The "Work Order Number" may be obtained from the user's supervisor. In general it will be the same number that appears in columns 39-48 of the IDENT control card. The project code (cols. 33-34) must also be given in this field.

The user must certify that this is official NASA business with his signature under 'certification'.

Use the current date for 'Date of Request'.

For 'Date Required' use the date when the cards will be needed, or used ASAP (as soon as possible).

'Security Classification' is self-explanatory.

For 'Upon Completion Notify' use the user's name, telephone number, and organization symbol. Also, if the user wishes his data mailed to him he must specify his mailing point.

To identify the user's data give a brief description as shown.

It is not necessary to specify "Type of Cards". Standard manilla cards are the only style currently available.

1.3.1 COMPUTER WORK REQUEST AND AUXILLIARY EQUIPMENT WORK REQUESTS (Cont.)

Mark the appropriate boxes for card processing services to be performed as this form is also used for listing, reproducing, interpreting, sorting, and collating of decks.

There are four character sets available to the programmer:

(1) IMBF (IBM FORTRAN); (2) IBMC (IBM COBOL); (3) IBMEL (IBM Extended Language); and (4) GE (General Electric).

Numbers (0-9) and letters (A-Z) are punched the same way in all four character sets, but the punches for special characters (+, @, #, etc.) may vary from set to set. IBMF is the standard character set at this installation, and the programmer's cards will be punched in this set unless he specifies otherwise on his Card ProcessingRequest form under 'Other Instructions'. An asterisk should be placed by the box for Keypunch and by this note so that the keypunch operator will be sure to notice the character set change.

Programmers always use the standard character set (IBMF) unless they need a special print character not included in this set.

Before running the job on the GE-635, the programmer must place a $ INCODE control card before a program deck or a data deck punched in any character set other than the GE set. This card tells the computer which set the cards were punched in the forces a transliteration from that set the GE set in memory. (This card is not used for decks punched in the GE character set). The format of this card is:

COL.	1	8	16
	$	INCODE	IBMF IBMC or IBMEL

Transliteration ceases when the next control card is encountered. Note that the special characters used in all control cards have the same card punches in all four character sets. (See Section 3.3 for examples of control Cards).

Figure 1.4 shows the results of special character transliterations for the various sets as they appear in GE-635 memory and on the printer.

CARD PROCESSING REQUEST

WORK ORDER NUMBER: 7PB0080000-20

TO:

FROM: *(Originator's symbol)*

CERTIFICATION: This work is requested for the conduct of official NASA business.

Your Signature Here
ORIGINATOR'S SIGNATURE

DATE OF REQUEST	DATE REQUIRED	SECURITY CLASSIFICATION
Nov. 11, 1970	ASAP	UNCLASSIFIED

UPON COMPLETION NOTIFY *(Name, telephone number and organization symbol)*

G. D. POE 7-8470 FEC 460

IDENTIFICATION OF MATERIAL

SHEETS, ALL PURPOSE TRANSCRIPT WITH FORTRAN CODING.

☒ PUNCH CARDS	☒ VERIFY	TYPE OF CARDS
		5081

☐ LISTING ☐ 80/80 ☐ DOUBLE SPACE ☐ SINGLE SPACE

NUMBER OF COPIES	TYPE OF PAPER	PANEL NAME/NUMBER

☐ REPRODUCE CARDS ☐ 80/80 ☐ SEQUENTIALLY NUMBER ☐ END PRINT

☐ OTHER	START	INCREMENT	COLUMNS

FROM	TO	FROM	TO	☐ GANG PUNCH	
				DATA	COLUMNS
1.		5.			
2.		6.		1.	
3.		7.		2.	
4.		8.		3.	

☐ INTERPRET ☐ UPPER 1-60 ☐ LOWER 61-80

☐ SORT ☐ COLLATE

MODE *(Circle one)*	COLUMNS	MODE *(Circle one)*	COLUMNS	SEQUENCE CHECK	MATCH	MERGE	SELECT
MINOR A A/N N		MINOR A A/N N					
INT. A A/N N		INT. A A/N N					
MAJOR A A/N N		MAJOR A A/N N					

OTHER INSTRUCTIONS

FIG. 1.3

RECEIPT OF COMPLETED JOB *(Signature, date, organization symbol)*

USE OF SPECIAL CHARACTERS ON GE-635

CARD PUNCH	RESULT OF IBM TRANSLITERATION OCTAL CODE	PRINT CHAR.	RESULT OF NO TRANSLITERATION (GE) OCTAL CODE	PRINT CHAR.	RESULT OF IBMEL TRANSLITERATION OCTAL CODE	PRINT CHAR.	RESULT OF IBMC TRANSLITERATION OCTAL CODE	PRINT CHAR.
2-8	12	[	12	[	15	:	12	[
3-8	75	=	13	#	13	#	75	=
4-8	57	'	14	@	14	@	76	"
5-8	13	#	15	:	57	'	13	#
6-8	16	>	16	>	75	=	16	>
7-8	17	?	17	?	76	"	17	?
12	60	+	32	&	32	&	60	+
12-3-8	33	.	33	.	33	.	33	.
12-4-8	55	]	34	]	36	<	55	]
12-5-8	35	(	35	(	35	(	35	(
12-6-8	36	<	36	<	60	+	36	<
12-7-8	37		37		37		37	
11-0	40	↑	40	↑	40	↑	40	↑
11	52	-	52	-	52	-	52	-
11-3-8	53	$	53	$	53	$	53	$
11-4-8	54	*	54	*	54	*	54	*
11-5-8	55	)	55	)	55	)	55	)
11-6-8	56	;	56	;	56	;	56	;
11-7-8	57	¬	57	¬	20		57	¬
12-0	60	+	60	+	60	+	60	+
0-1	61	/	61	/	61	/	61	/
0-2-8	72	←	72	←	20		72	←
0-3-8	73	,	73	,	73	,	73	,
0-4-8	35	(	74	%	74	%	35	(
0-5-8	75	=	75	=	20		75	=
0-6-8	76	"	76	"	16	>	76	"
0-7-8	77	!	77	!	17	?	12	[

FIG. 1.4

MEMORY REP. (OCTAL)	PRINTER CHAR.	CARD PUNCH		MEMORY REP. (OCTAL)	PRINTER CHAR.	CARD PUNCH	
		GE	IBMF			GE	IBMF
00	0	0	0	40	↑	11-0	11-0
01	1	1	1	41	J	11-1	11-1
02	2	2	2	42	K	11-2	11-2
03	3	3	3	43	L	11-3	11-3
04	4	4	4	44	M	11-4	11-4
05	5	5	5	45	N	11-5	11-5
06	6	6	6	46	Ø	11-6	11-6
07	7	7	7	47	P	11-7	11-7
10	8	8	8	50	Q	11-8	11-8
11	9	9	9	51	R	11-9	11-9
12	[	2-8	2-8	52	-	11	11
13	#	3-8	5-8	53	$	11-3-8	11-3-8
14	@	4-8	None	54	*	11-4-8	11-4-8
15	:	5-8	None	55	)	11-5-8	11-5-8
16	>	6-8	6-8	56	;	11-6-8	11-6-8
17	?	7-8	7-8	57	'	11-7-8	11-7-8
20	blank	Space	Space	60	+	12-0	12
21	A	12-1	12-1	61	/	0-1	0-1
22	B	12-2	12-2	62	S	0-2	0-2
23	C	12-3	12-3	63	T	0-3	0-3
24	D	12-4	12-4	64	U	0-4	0-4
25	E	12-5	12-5	65	V	0-5	0-5
26	F	12-6	12-6	66	W	0-6	0-6
27	G	12-7	12-7	67	X	0-7	0-7
30	H	12-8	12-8	70	Y	0-8	0-8
31	I	12-9	12-9	71	Z	0-9	0-9
32	&	12	None	72	←	0-2-8	0-2-8
33	.	12-3-8	12-3-8	73	,	0-3-8	0-3-8
34	]	12-4-8	None	74	%	0-4-8	None
35	(	12-5-8	12-5-8	75	=	0-5-8	3-8
36	<	12-6-8	12-6-8	76	"	0-6-8	0-6-8
37	\	12-7-8	12-7-8	77	!	0-7-8	0-7-8

FIG. 1.5

1.3.1 COMPUTER WORK REQUEST AND AUXILLIARY EQUIPMENT WORK REQUESTS (Cont.)

Figure 1.5 shows each character as it appears in memory and on the printer for the GE and IBMF character sets.

GE-635 Work Requests

Sample Computer Operations Work Requests are show in Figure 1.4 and 1.5. As with the Card Processing Request the submitted job should have with it three copies, one original and two duplicates. The operator at the I/O counter will take the user's deck, check for an $ IDENT control card, $ ENDJOB control card, and an ***EOF control card. If the deck is in order a $ SNUMB control card will be placed in front of the deck and the corresponding $ SNUMB number will be written on the user's Request form. In the example, S-21758 is the SNUMB. This number will stay with the job until completion. It will appear on all listings and decks generated by the user's program. After the number has been written on the top of the Request sheet the operator will initial it in the "Received By" column and return the second copy.

When the job is completed the Request form will be initialled by the Operator and marked complete in the job column. The user will initial the Request form in the "Received By" column and receive the first copy. The original is kept for job accounting purpose.

The 'Classification' the user will probably circle is "U" for unclassified.

There are many levels of 'Priority' but the two that most Programmers use are 300 for compiles, and 700 for test runs. Do not use any other priorities unless specifically instructed by the Supervisor.

Checkmark which machine is being requested (635) and initial the 'Received From' column.

The 'Program', 'Control No.', and 'Job Identification' fields are taken from corresponding fields on the $ IDENT control card used for this submittal. The user should compare these fields with their counterparts in the sample $ IDENT control card described later in this manual. Figure 1.6 and Figure 1.7 are sample GE-635 work requests.

COMPUTER OPERATIONS WORK REQUEST

S-21758

CLASSIFICATION	PRIORITY
U C S	700

TIME IN	TIME OUT	REQUESTOR'S NAME	PHONE
		G. POE	7-8470

RECEIVED FROM	RECEIVED BY	JOB	√
		KEYPUNCH AND VERIFY	
		EAM	
		1004	
		7702	
		635	X
		4020	

JOB NO.	PROGRAM	CONTROL NO.
	AVER1	SP1000

ITEMS SUBMITTED TO BE RETURNED

DESCRIPTION	SECURITY CLASS	COPIES SUBMT.
OBJECT DECK	U	1
DATA DECK	U	1

INPUT TAPES

REEL	CONTROL	UNIT
Reel	#8113	

OUTPUT TAPES

REEL	CONTROL	UNIT
Reel #7341		
SCRATCH		

ITEMS TO BE GENERATED

DESCRIPTION	SECURITY CLASS	COPIES REQUEST	COPIES GENERATE
ACTIVITY 1 - Listing	U	1	

RERUN CODES

1	2	3	4	5	6	7	8	9	A	B	C	D	E

JOB IDENTIFICATION

7PB0080000 - 20

REMARKS:

9 K

.1 Hrs.

3 Tapes

FIG. 1.6

KSC FORM 22-53 (REV 3/66)

NASA-PAFB MAY/71

COMPUTER OPERATIONS WORK REQUEST

S-21758

CLASSIFICATION	PRIORITY
U C S	700

TIME IN	TIME OUT	REQUESTOR'S NAME	PHONE
		G. POE	7-8470

RECEIVED FROM	RECEIVED BY	JOB	√
		KEYPUNCH AND VERIFY	
		EAM	
		1004	
		7702	
		635	X
		4020	

JOB NO.	PROGRAM	CONTROL NO.
	AVER1	SP1000

ITEMS SUBMITTED TO BE RETURNED

DESCRIPTION	SECURITY CLASS	COPIES SUBMT.
FORTRAN DECK	U	1

INPUT TAPES				OUTPUT TAPES		
REEL	CONTROL	UNIT		REEL	CONTROL	UNIT

ITEMS TO BE GENERATED

DESCRIPTION	SECURITY CLASS	COPIES REQUEST	COPIES GENERATE
ACTIVITY 1 - Listing	U	1	
Deck	U	1	
ACTIVITY 2 - Listing	U	1	

RERUN CODES

1	2	3	4	5	6	7	8	9	A	B	C	D	E

JOB IDENTIFICATION

7PB0080000 - 20

REMARKS:

9 K

.005 Hrs.

No Tapes

FIG. 1.7

23

1.3.1 COMPUTER WORK REQUEST AND AUXILLIARY EQUIPMENT WORK REQUESTS (Cont.)

GE-635 Work Requests (Cont.)

The 'Input Tapes' and 'Output Tapes' information also is taken from control cards that the user will have prepared if tapes are needed for this submittal. This will be discussed in more detail when the $ TAPE control card is described later in this manual. See Fig. II.3.

In the "Remarks" field of the Request Form some information is required to help the Operations personnel use the multiprogramming capabilities of the 635 efficiently. The data needed is: (1) an estimate of the run time, (2) an estimate of the core storage, and (3) the number of tapes required.

For compiles only, estimate 10 minutes, 16K, and "No Tapes" and this will be close enough for the purpose.

For execution of programs take the core limitation on the $ LIMITS card. Once again this will be discussed in greater detail when the $ LIMITS control card is described later in this manual. Time is a much more difficult estimate, but executions with sorts, a lot of printing, or a great deal of calculations will take longer. A simple Fortran execution will generally take very little time and 5 minutes should be sufficient.

1.3.2 PRIORITY SYSTEM FOR GE-635

The following priority system is used for Computer Operations. This system applies to the scientific computer room 205 only.

CODE	EXPLANATION
0 - 099	Zero type priority of the highest that can be used. It is reserved for real-time test support on all vehicles. This will be primarily a data reduction priority.
100 - 199	This priority is primarily used for schedules data core or display testing. It may also be used for retrieving the post-test analog data. This priority may be used by both programming and data reduction personnel.

1.3.2 PRIORITY SYSTEM FOR GE-635 (Cont.)

CODE	EXPLANATION
200 - 299	This priority is reserved for the commercial printing that needs to be done on the scientific computers. Basically, this priority is limited to one printer on each computer.
300 - 399	This priority is used by the programmer for program compilation and/or assemblies only. There is to be no program test work done under this priority.
400 - 499	Critical post-test data-reduction functions will fall in this category. This priority will be used only for data with a 4-hour or less requirement time. This priority will be noted on the data request.
500 - 599	This priority is used by the programmer for program testing on jobs of a critical nature. The total number of these priorities is not to exceed 10 per cent of the total number of programs (examples: if there is a total of 200 programs being written, only 20 can have this priority). This priority will be designated in advance to the programmer by his supervisor.
600 - 699	All other post-test or post-launch data reduction requirements will fall in this category. Computer Operations is to limit the primary data reduction effort to one GE-635. If there is machine time available on the second GE-635 system, Computer Operations would be allowed to process data reduction jobs that require less than 30 minutes processor time. This implies that data reduction would never have the second system tied up more than 30 minutes.
700 - 799	This priority is used for all other program testing and data reduction work. These jobs will be processed according to the time received at the input/output counter.
800	Training runs. All personnel.

1.4 MANUALS

1.4.1 GE-635 MANUALS

CPB NO.	TITLE OF MANUAL
CPB3798	GE-625/635 REFERENCE CARD
CPB380A	GE-625/635 MNEMONICS AND OPERATION CODES-REFERENCE CARD
CPB416A	GE-600 REFERENCE CARD
CPB481B	INTEGRATED DATA STORE-DATA BASE STUDY
CPB1003F	GE-625/635 FILE AND RECORD CONTROL
CPB1004F	GE-625/635 PROGRAMMING REFERENCE MANUAL
CPB1005C	GE-625/635 SORT/MERGE PROGRAM
CPB1006F	GE-625/635 FORTRAN IV
CPB1008E	GE-625/635 GENERAL LOADER
CPB1077A	GE-625/635 GIFT GENERAL INTERNAL FORTRAN TRANS.
CPB1078B	GE-600 SERIES GMAP IMPLEMENTATION
CPB1080A	GE-625/635 SYSTEM EDITOR
CPB1081A	GE-625/635 IBM 7094 SIMULATOR
CPB1083B	GE-625/635 FORTRAN IV MATH LIBRARY
CPB1087X	GE-625/635 ALGOL
CPB1093B	GE-625/635 INTEGRATED DATA STORE
CPB1096D	GE-625/635 BULK MEDIA CONVERSION
CPB1126	GE-625/635 GEFRC IMPLEMENTATION
CPB1127	GE-600 SERIES GELOAD GENERAL LOADER
CPB1130	FORTRAN IV-SSI-600
CPB1133A	GE-625/635 INTEGRATED DATA STORE-SSI
CPB1137	FORTRAN I/O LIBRARY-SSI
CPB1138A	GE-625/635 SYSTEM EDITOR MANUAL
CPB1139A	GE-625/635 PER/TIME
CPB1141B	GE-600 INTRODUCTION TO LINEAR PROGRAMMING LP/600
CPB1152A	GE-625/635 POLRTS MATH ROUTINES
CPB1165	GE-600 SEQUENTIAL LEAST-SQUARE FOR POLYNOMIALS
CPB1166	GE-600 EIGEN-VALUES/-VECTORS OF SYMETRIC MATRIX
CPB1167	GE-600 SIMEQ-SIMULTANEOUS LINEAR EQUATIONS
CPB1168	GE-600 DIFFE-DIFFERENTIAL EQUATION SOLVER
CPB1169A	GE-600 NUMERIDRILL-APT POSTPOCESSOR
CPB1171	GE-600 APT-POST PROCESSOR FOR A MONARCH LATHES ETC.
CPB1175	GE-600 GECENT BURGMASTER APT POSTPROCESSOR
CPB1176	GE-600 GECENT MULWAUKEE-MATIC APT POSTPROCESSOR
CPB1183	GE-600 BMD BIOMEDICAL STATISTICAL PROGRAMS
CPB1184	GE-600 1401 SIMULATOR REFERENCE MANUAL
CPB1185	SOFTWARE LIBRARY REFERENCE MANUAL
CPB1186	GE-600 225 SIMULATOR
CPB1187AX	GE-600 JOVIAL REFERENCE MANUAL
CPB1188	GE- 625/635 IBM 7040/44 SIMULATOR
CPB1192A	GE-625/635 PER/TIME-SSI

1.4.1 GE-635 MANUALS

CPB NO.	TITLE OF MANUAL
CPB1195A	GE-625/635 COMPREHENSIVE OPERATING SUPERVISOR
CPB1200	GE-625 APT POSTPROCESSOR BULLARD
CPB1201AX	GE-625/635 JOVIAL COMPILER REFERENCE MANUAL
CPB1218X	GE-625/635 SIMSCRIPT COMPILER
CPB1222	GE-625/635 LP/600 INPUT FILE PREPARATION
CPB1228	GE-625/635 225 SIMULATOR-SSI
CPB1238A	GE-200/400/600 TIME SERIES FORECASTING
CPB1262	GE-625/635 LP/600 AGENDA CONTROL LANGUAGE
CPB1263	GE-625/635 LP/600 MATRIX GENERATOR LANGUAGE MANUAL
CPB1264	GE-625/635 LP/600 FORMAT GENERATOR LANGUAGE
CPB1267	GE-625/635 LP/600 OUTPUT DESCRIPTIONS
CPB1269	GE-400/600 SERIES TRAVERSE ANALYSIS SYSTEM-CEP
CPB1270	GE-400/600 SERIES CURVED BRIDGE GEOMETRY-CEP
CPB1271	GE-400/600 SERIES SLOPE STABILITY ANALYSIS-CEP
CPB1272X	GE-400/600 CEP RETAINING WALL DESIGN
CPB1273	GE-400/600 SERIES CONTINUOUS GIRDER ANALYSIS-CEP
CPB1274X	GE-400/600 CEP COMPOSITE BEAM ANALYSIS
CPB1275X	GE-400/600 HORIZONTAL GEOMETRY PROGRAM SYSTEM
CPB1276X	GE-400/600 SERIES EARTHWORK PACKAGE
CPB1321X	GE-400/600 STRUCTURAL ANALYSIS SYSTEM SUPER
CPB1376X	GE-200 GECOM TO 400/600 COBOL TRANSLATOR-GETRAN
CPB1384X	GE-625/635 PER/COST
CPB1417A	GE-625/635 GERTS II PROGRAMMER'S REFERENCE MANUAL
CPB1422A	GE-625/635 UTILITY
CPB1477A	GE-625/635 TYPEWRITER MESSAGES GECOS-II
CPB1488	GE-625/635 GECOS III INTRODUCTION AND SYSTEM TABLES
CPB1489	GE-625/635 GECOS III STARTUP
CPB1490	GE-625/635 GECOS III SYSTEM INPUT
CPB1491	GE-625/635 GECOS III DISPATCHER AND PERIPHERAL ALLOCATION
CPB1492	GE-625/635 GECOS III R.C., CORE ALLOCAT., OP. INTERFACE
CPB1493	GE-625/635 GECOS III FAULT PROCESSING & SERVICE MME'S
CPB1494	GE-625/635 GECOS III I/O SUPERVISION
CPB1495	GE-625/635 GECOS III EXCEPTION PROCESSING
CPB1496	GE-625/635 GECOS III TERMINATION AND SYSTEM OUT OUTPUT
CPB1497	GE-625/635 GECOS III FILE SYSTEM MAINTENANCE
CPB1498	GE-625/635 GECOS III FILE SYSTEM MAINTENANCE
CPB1498	GE-625/635 GECOS III UTILITY ROUTINES

1.4.1 GE-635 MANUALS

CPB NO.	TITLE OF MANUAL
CPB1500	GE-625/635 GECOS III FLOWCHARTS
CPB1501	GE-625/635 GECOS III TIME-SHARING SYSTEM
CPB1509	GE-600 SERIES 615 SUPPLEMENT
CPB1510	GE-625/635 GECOS III TIME-SHARING SYSTEM-BASIC LANGUAGE
CPB1513	GE-625/635 GECOS III FILE SYSTEM
CPB1514	GE-625/635 GECOS III TIME-SHARING SYS.-PROGRAMMING REF.
CPB1515	GE-625/635 TIME-SHARING SYSTEM
CPB1518	GE-625/635 GECOS III COMPREHENSIVE OPERATING SUPERVISOR
CPB1562	GE-625/635 COBOL REFERENCE MANUAL
CPB1563	GE-625/635 COBOL PROGRAMMING GUIDE
CPB371D	625/635 SYSTEM MANUAL
CPB1044B	MAGNETIC TAPE SUBSYSTEM - 600
CPB1045B	GE-625/635 OPERATOR'S REFERENCE MANUAL
CPB1106A	PS-6010 PROGRAMMED PERIPHERAL SWITCH
CPB1110B	PRT-200 PRINTER
CPB1120A	PTS-200-PERFORATED TAPE SUBSYSTEM
CPB1153B	MDS-200 MAGNETIC DRUM SUBSYSTEM
CPB1205	SEVEN/NINE TRACK MAGNETIC TAPE SUBSYSTEM
CPB1288A	GE-400/600 SERIES PUNCHED CARD SUBSYSTEMS
CPB1292A	PRT-201 PRINTER
CPB1302A	DSU-200 DISC STORAGE SUBSYSTEM REFERENCE MANUAL
CPB1306A	GE-625/635 OPERATORS HANDBOOK
CPB1313X	PRT-202 PRINTER
CPB1416	GE-625/635 GERTS REFERENCE MANUAL
CPB1418A	GE-625/635 GERTS OPERATOR REFERENCE MANUAL
CPB1419	DISC STORAGE SUBSYSTEM-DSU 270
CPB1469	GE-625/635 TYPEWRITER MESSAGES-GECOS II
CPB1477	GE-625/635 GECOS III TYPEWRITER MESSAGES
CPB1177X	GE-625/635 APT III SYSTEM OPERATIONS MANUAL
CPB1179A	GE-625/635 APT III EXECUTIVE CONTROL-SSI
CPB1277	GE-625/635 APT III INPUT TRANSLATOR-SSI
CPB1278	GE-625/635 APT III ARELM
CPB1279	GE-625/635 APT III CUTTER LOCATIONS PROCESSOR-SSI
CPB1280	GE-625/635 APT III POSTPROCESSOR DISPATCHER-SSI
CPB1281	GE-625/635 APT III INTERNAL FILE FORMAT-SSI
CPB1283A	GE-625/635 APT III GEOMETRICS ROUTINE
CPB1309	GE-625/635 APT III SECTION 1 DIAGNOSTICS-SSI
CPB1423	GE-625/635 APT III COMPOSITE INDEX-SSI
600-142	CORRECTION TO CPB 1280
600-169	CORRECTION TO CPB 1281
600-170	CORRECTION TO CPB 1309
600-180	CORRECTION TO CPB 1277

1.4.2 UNIVAC 1005 COMPATIBLE MANUALS

UP-7529	80/90 Condenser
UP-4084	Programmers Reference
UP-4088	Report Program Generator
UP-4072	Utility Programs
UP-4089	Assembler
UP-4052	System General Description

1.4.3 S-C 4020 PLOTTER MANUALS

S-C 4020 Computer Recorder. DOC. NO. 950056
GE-635 SC-4020 SUBPAK (No Origin)

1.4.4 TIME-SHARING MANUALS

GENERAL INFORMATION	CPB-1643
FORTRAN	CPB-1566
BASIC	CPB-1510
TERMINAL BATCH INTERFACE	BR-99 (Formerly CPB-1642)
TEXT EDITOR	CPB-1515

SECTION 2 PROGRAMMING STANDARDS AND PROCEDURES

2.1 STANDARD PRODUCTION PROGRAM PROCEDURES

2.1.1 MANDATORY REQUIREMENTS

The sub-routines and procedures listed below are pre-requisite to all production slave programs:

Procedure

DBUG1 - De Bug
CKID - Check ID
GEIN & Sysout
Parity Error Routine
BMC - Sysout Options
Listing Input Cards
Unit Switching
Multi-file Reel
Disc Usage For Scratch File
Magnetic Tape Label Processing
EOF - End of File

Although the control cards for the above items are not supplied by programming, familiarization with their format is of significant value. These cards and their formats are listed in the GE Manual CPB-1688.

DBUG1 SUBROUTINE

Abstract:

There are occasions that require the processing of data by a program before the program has been completely checked out and placed in production. These occasions occur when test data is not available prior to a test to checkout a program, or when a program has been placed in production and a bug has been detected and a correction made. When this occurs, programming personnel normally run the program. It has been necessary in the past that data resulting from these runs be marked as "test data" before being issued. The DBUG1 subroutine provides an automatic means of having the "test data" message printed on the ourput, and should become a standard feature for all programs producing printed data.

2.1.1 MANDATORY REQUIREMENTS (Cont.)

Function:

When called by the appropriate calling sequence, DBUG1 examines the CKID test word for a "D" in the least significant character. If the character "D" is present, the subroutine then moves a 132 character print line containing the appropriate print message to the 22 computer word print line indicated by the second argument of the calling sequence. If the least significant character of the CKID test word is not a "D" (octal 24), then 132 characters of blanks are stored in the designated print line. The operation of DBUG1 is dependent upon the CKID subroutine, which is also a programming standard.

The CKID subroutine, when called, will examine the program ID, as found on the CKID card, and compare it with the contents of the CKID test word specified in the calling sequence, as it normally does. If the compare is successful, CKID will also test columns 6-10 of the CKID card for the word DEBUG. If present, CKID will place a "D" (octal 24) in the least significant character of the CKID test word. It is this information for which the DBUG1 subroutine tests to determine whether to transfer the "test data" message or to transfer blanks.

USAGE:

To use DBUG1, columns 6-10 of the CKID card must contain DEBUG if the test data message is to be printed. If anything other than these characters appear in columns 6-10, then no test message will appear.

CHECK ID:

Purpose

To insure that the input data deck matches the program deck by ID number.

Usage:

Calling Sequences - CALL CKID (A) CALL CKID (A,B,)
Where A = Location of five-character program ID, left justified.*
B = Location of the File Control Block to be used to read in the ID card.
CKID uses 517 words.
No error conditions.

2.1.1 MANDATORY REQUIREMENTS (Cont.)

*If the word DEBUG is used in this location it will call up the DBUG1 subroutine.

Restrictions

None.

Method

Reads in ID card, prints columns 6 through 80 and compares the card ID, columns 1 through 5, to the internally starred program ID.

If the card ID and the program ID do not match, the following message is written on SYSOUT and the program is aborted with a CK abort code. (MME GEBORT).
PROGRAM ID (XXXXX) DOES NOT CHECK WITH CARD ID (XXXXX).

If only one argument is used, the subroutine checks to see whether a $ OPTION FORTRAN (or FCB) control card was used (cell 25_8 1 0). If so, the location of CKID's FCB for I* is started in the FORTRAN FCB list. Then, in either case, CKID's I* FCB is opened and GET is used to read the ID card.

The ID information is written on SYSOUT using CKID's P* (Execution report code) using IOEDIT and PRINT. The location of this FCB is not stored in the FORTRAN FCB list.

USE OF GEIN AND SYSOUT

GEIN is an established system file (I*), designed for low-volume card input. GEIN may be used for input of less than 100 cards. In all other cases BMC should be used for card-to-tape or card-to-disc functions. Data cards destined for GEIN should be placed between the last $ file card and the $ ENDJOB cards.

SYSOUT is the system file (P*) for low-volume print and/or punch output. The limits of the current tape SYSOUT are 10,000 print lines and/or 1000 punch cards. Data may be directed to SYSOUT by the DISPLAY verb in COBOL, or by using the $ SYSOUT file card. SYSOUT should be used by programmers primarily for test output. SYSOUT should not be used for the printing of scheduled reports. This is based on the NASA requirement that report tapes must be retained a

2.1.1 MANDATORY REQUIREMENTS (Cont.)

minimum of five days. In addition, custom forms and multi-paper cannot be used on SYSOUT.

PARITY ERROR PROCEDURE

All programs which read input tapes should allow for the possibility of unrecoverably parity errors. It should be determined by the analyst whether the program should ignore the error and process the data record normally or skip the record completely. In either event the program should give a message at the end-of-run stating the number of unrecoverable parity errors.

BMC-SYSOUT OPTIONS

Programs having a substantial amount of printer output should, unless otherwise requested, allow for an option to put the output on a BMC tape instead of SYSOUT. This will allow more efficient utilization of the printer during production runs.

LISTING INPUT CARDS

All input cards should be printed at the beginning of each run. Sufficient labels should be supplied, giving an analyst or engineer as much information as possible to assist him in evaluating his data.

UNIT SWITCHING PROCEDURE

Programs having multi-reel input should have a clearly defined procedure for unit switching. If GEINOS is used, mounting and dismounting instructions to the operator should be issued on the typewriter. If GEFRC is used and the $ Tape card is correct, the system will execute unit switching automatically.

2.1.1 MANDATORY REQUIREMENTS (Cont.)

MULTI-FILE REEL PROCEDURE

Some input tapes such as trajectory tapes have more than one file on a single reel of tape. The programmer should determine if his input tape is multi-file and if so, design his program to input the number of files to process from punched card or etc. In Fortran a useful subroutine in processing multi-file reels is FLGEØF. (See memo FLGEØF in this section.)

DISC USAGE FOR SCRATCH FILES

When it becomes necessary for a programmer to use several scratch tapes, investigating the use of DISC might prove beneficial. This could possibly improve turn around time and also the added reliability feature of the DISC over tape.

MAGNETIC TAPE LABEL PROCESSING

GEFRC contains a label-processing capability. Two routines are available that will inhibit label processing when labels do not appear on tape.

.GOPNB	routine to turn off labels on all files opened.
.GOPNA	routine to turn off labels on all input files opened.

The following example will illustrate the use of these routines.

```
$       IDENT
$       USE             .GOPNB
$       OBJECT
```

When label processing is enabled all that is necessary is the removal of the $ USE Card.

2.1.1 MANDATORY REQUIREMENTS (Cont.)

END OF FILE SUBROUTINE

All programs which read tape and/or cards should have an End of File Subroutine. It should never be assumed that a program will always terminate before reaching and end-of-file mark even though the program algorithm indicates it should.. A bad input tape or a control card error could cause the program to run to end-of-file.

It is possible, during recording, for "noise" to be written as single character records on data tape. If GEFRC is used to read the tape and it is unlabeled, any single character record will be interpreted as an end-of-file mark, and control will go to the user's EOF Subroutine. It is the user's responsibility to check for a specific EOF mark such as an Octal 17. (See GE-635 File and Record Control Manual, pp.79.)

SECTION 2 PROGRAMMING STANDARDS AND PROCEDURES

2.2 GENERAL

2.2.1 TIME EDIT PROCEDURE

In programs where time is an input variable, it may be necessary to identify time discrepancies. Binary bits may be added or dropped from time words to give distorted time values. Care should be taken not to confuse distorted time with legitimate time discontinuity such as time recycle and time dropout. Time recycle occurs when the time goes over midnight, Greenwich mean time, to 00.00 hours (i.e., 23 hrs., 59 mins., 59 secs., to 00.00 hrs., 00 mins., 00 secs.) Time dropout may occur when the recording machinery is shut down for a period of time for which there are no data requirements. Unless these conditions are anticipated, the program may confuse time recycle and time dropout with distorted time, For further information and suggestions on recovery, see Timing Problems on Recorded Telemetry Data tapes and Suggested Recovery Techniques, October 12, 1966, NASA Technical Programs Section.

2.2.2 TESTING MODIFICATIONS

After a program has been modified, no matter how slightly, the programmer should make at least one (1) test run.

2.2.3 COMPRESSED DATA PROGRAMS

All compressed data programs should contain a message indicating the time span searched when no data is output. As many checks as possible should be made on input data especially where it is being used to set up an indexing scheme. This can prevent F0 or F8 aborts in the future.

2.2.4 SYMBOLIC FILE CODES

Each file referenced in a 635 program must be identified by a two-character alphanumeric file code. The file code may be either two letters (A-Z) or a letter and a digit (0-9). The following restrictions are placed upon the construction of a file code:

2.2.4 SYMBOLIC FILE CODES (Cont.)

The letter S may not be used.

File codes IN and OT may not be used.

Programmers must specify the file codes in the FILCB macro in a GMAP program.

File codes used in a program must also be used as the first parameter (column 16) of the $ file cards ($ TAPE, $ DISC, etc). The system can then relate the file referenced in a peripheral units.

Programmers should be aware of the fact that the file code in the program and the file code in the $ FILE card need by unique only for a particular activity. If a file is created in one activity, and read in another, the file codes may be different: then only the channel and unit designators need be identical.

```
First Activity                  $ TAPE QR, A2S,,1,, MASTER
Second Activity                 $ TAPE AB, A2R
```

would refer to the same physical tape.

2.3 I/O FILE AND RECORDS

2.3.1 MAGNETIC TAPE

Unit Switching

The method used for fibs containing two or more reels of tape. The tapes are mounted alternately on two tape units. The $ TAPE control card is used with columns 23-25 containing a file code designating the second tape. (See Section 3.3, Control Cards.)

Unit Switching On Multi-File Tapes With Non-Standard Labels

Some problems have occured in connection with unit switching in FORTRAN and GEFRC with multi-file tapes with non-standard labels. As a result of investigations involving these problems, the attached method was found to be the simplest and at the same time surest way to perform unit switching on multi-file tapes with non-standard labels. Note that this is a subroutine which may be called by FORTRAN but the CALL FORCE coding should be used in any situation in a GMAP routine where such unit switching is desired.

```
*                       ROUTINE TO UNIT SWITCH NON-LABELED
*                       MULTIFILE TAPES FOR FORTRAN USING A
*                       READ (10) STATEMENT TO DO THE READING
*                       IN THIS EXAMPLE.
       SYMDEF   TYPR
*                       THE SYMREF TO .FBAD. CARD MUST BE
                        PRESENT.
       SYMREF   .FBAD.
TYPR   SAVE     0,1,2,3,4,5,6,7
*                       STORE FILE CODE SO THAT GEFRC CAN FIND
*                       THE LOCATION SYMBOL OF THE PROPER FILE
*                       CONTROL BLOCK WHERE 10 IS THE LOGICAL
*                       UNIT DESIGNATOR.
       LDA      =10,DL
       STA      .FBAD
*                       OPEN THE FILE CONTROL BLOCK AND PLACE
*                       THE LOCATION SYMBOL OF THE FILE CONTROL
*                       BLOCK IN THE .FBAD. CELL.  IF THE FILE
*                       IS ALREADY OPEN. THE CALL .FOPEN WILL
*                       CAUSE NO DIFFICULTY.
       CALL     .FOPEN (=-1)
*                       STORE THE LOCATION SYMBOL IN THE SUB-
*                       ROUTINE ARGUMENT.
       LDX3     .FBAD
       STX3     CALL1+3
*                       SWITCH MAGNETIC TAPE LOGICAL UNIT.
CALL1  CALL     FORCE(**)
       RETURN
       END
```

2.3.1 MAGNETIC TAPE (Cont.)

TAPE USAGE AND RETENTION

Tape Retention Period

Due to the increasing number of tapes being catalogued and the limited storage area available, a time limit has been set to determine the retention period. If it becomes necessary to retain a tape beyond this period, notify the tape librarian any time before the scratch date. Computer operations will distribute on a weekly basis, a list of all tapes that have reached their scratch dates. This list will serve as a reminder to the Programming and Data Reduction personnel.

The types of tapes and their retention period will be as follows:

a.	BMC	2 Days
b.	Intermediary (PITT, Plot input, etc.)	2 Days
c.	Digital Balloon	4 Days
d.	Plot	7 Days
e.	Transceive	7 Days
f.	Debug and Test	30 Days
g.	Coefficient	One (1) Test or 100 Days
h.	Raw Data	One (1) Test or 100 Days
i.	Trajectory	One (1) Launch plus 7 Days
j.	MSC Calibration	100 Days
k.	Training Aids	30 Days
l.	SCO Compressed Tapes	15 Days
m.	SCO Descriptor or Check Tapes	100 Days

All magnetic tapes that are to be retained should be logged into the tape library. This will avoid any possible damage due to temperature and humidity effects during delivery and storage out of the controlled area. Any tape that is not to be retained should be clearly identified as a scratch tape on the computer request form.

2.3.2 DISC

Disc Usage for Scratch Files.

When it is necessary to use several scratch tapes, investigating the use of DISC could be beneficial to improve turn-around time and the additional reliability of DISC

2.3.2 DISC (Cont.)

over Tape. This is accomplished by simply removeing the $ TAPE card(s) and inserting $ DISC card(s).

Restrictions: All scratch files must use Standard System Format.

2.3.3 DRUM CAPABILITY

The programmer has access to 1,093,632 words of storage on the drums. He has available 2 methods of storing information on the drum. He can store it as a random or as a linked file. Random files are handled through CALL READ or CALL WRITE options of GEFRC. These files are referenced by a block number relative to the beginning of the file. Each block is 64 words long and may be grouped.

For linked files, standard systems format must be used and Fortran or GEFRC standard routines can be used. The storage reserved on the drum is done by a $ DRUM card, containing the number of links needed.

2.3.4 MULTI-FILE REEL PROCEDURE

Some input tapes such as trajectory tapes have more than one file on a single reel of tape. The programmer should determine if his input tape is multi-file and if so, design his program to input the number of files to process from punched card or etc. In Fortran a useful subroutine in processing multi-file reels is FLGEØF. (See memo FLGEØF in this section.)

2.3.5 MULTIPLE CONSOLE LOGIC

In conjunction with the introduction of multiple Console logic in GECOS, Module MXOO has been modified to handle the Master and Auxiliary Console typewriters as follows:

1. The real-time supervisor will always direct its system messages which are not related to operator requests to the MASTER Console.

2.3.5 MULTIPLE CONSOLE LOGIC (Cont.)

2. The real-time slave program may direct his request for an operator initiated courtesy call (RTSPEC) a particular Console.

3. Depressing the request key on any Console while in real-time mode will cause the real-time supervisor to respond. The response issued is RT???

4. In responding to operator requests, the real-time supervisor will direct its questions and responses to the Console from which the operator request was initiated.

5. The Master system Console and all Consoles allocated to the real-time slave program are shared with the batch processing supervisor and its slaves.

6. Special interrupts from the shared Consoles are passed on to the batch processing supervisor only after the operator has been queried and only if the response received does not pertain to real-time.

7. The file codes for the MASTER and AUXILIARY Consoles are T/ and /T respectively.

In conjunction with #2 above, the master mode entry RTSPEC has been modified to function as follows:

```
MME     RTSPEC
ZERO    CCARDS, FILPTR
return
```

When CCARDS in non-zero, it is taken as the first location of a courtesy call routine to which control will be transferred upon receipt of the operator input "RTSLAVE" from the pertinent Console.

When CCARDS is zero, the outstanding RTSPEC on pertinent Console will be cancelled. When the RTSPEC courtesy call is paid, the RTSPEC is automatically concelled.

When FILPTR is non-zero, it is taken as the location containing the file code of the Console to which the request is directed.

When FILPTR is zero, the request is automatically directed to the Master System Console.

2.3.6 DEFINITIONS OF GEFRC TERMS

File - a collection of records of data that may be referenced either sequentially (from first to last) or randomly (by the record number). It may be a set of cards in the card reader, a set of print lines on the printer, a set of records on a tape or disc, etc. In any case it lies between some type of end-of-file markers.

End-of-File Mark - on the GE635 is any single character record, by the standard file mark in GEFRC is an octal 17 character. Any request to position a device before or after a file mark (such as backspace to file mark etc.) will result in a search for a single character record of octal 17. When any other file mark is detected, control is passed to the user's end-of-file routine specified in the calling sequence used to reference the record.

File Control Block - an ordered block of data used to define the GEFRC routines the physical characteristics of the file obtained partially from the user's file control cards and partially be definition from the file control block macro-instruction and to indicate which of the various types of processing are to be used by this file.

The File Control Block Macro-Instruction - the method of initially defining the contents of the file control block for a given file. It must be a part of the coding of any program using GEFRC I/O routines, and there must be one block for each file in your program.

Working File Control Block - the contents of the file control block during program execution.

File Control Block Location Symbol (LOCSYM) - the address of word 0 of the working file control block in your program. Reference the GEFRC manual's section entitled "Working File Control Block Format".

Opening the File Control Block - refers to the process of (1) connecting the file control block to a physical device by means of the file code described below and (2) performing the optional tasks (such as rewind an open etc.) specified in the "File Designator Word" referenced in the CALL OPEN GEFRC entry.

Closing The File Control Block - refers to the process of (1) disconnecting the device from the file control block and (2) performing the optional tasks (such as rewind and lock on close) specified in the "File Designator Word" referenced by the CALL CLOSE GEFRC entry.

2.3.6 DEFINITIONS OF GEFRC TERMS

File Designator Word - a word used by the GEFRC OPEN and CLOSE routines that gives the address of the LOCSYM (word 0) of the file control block that you wish to open or close in bits 0-17 followed by a set of flags that determine what options you wish performed by the OPEN and CLOSE routines.

File Code - used in both file control cards and file control blocks. It is the means by which the physical device specified on the file control card is connected to the file control block within the program.

File Control Cards - the means by which the person putting a program into execution defines what type of device he wants associated with the file code (and thereby indirectly to the file control block in your program that contains the file code) designated on the file control card.

User's Error Processing Routines - a routine in your program which will be entered from GEFRC only after the GE Error Processing Routines have failed in all attempts to recover from an error. Its entry location is specified in your file control block macro-instruction, and if your do not specify such an entry, your program will be aborted whenever an unrecoverable error is encountered on the file.

Standard Labels - Labels at the beginning and end of each file on magnetic tape which uniquely identify the file by installation, date, retention period, reel serial no., reel sequence no., and file name. They are assumed for all tape files stated to be in "Standard System Format", but if they are not applicable to the device assigned to the file at execution, label processing is used, it is possible to protect files from being accidentally written over or using the wrong input tape.

Unit Switching - describes the procedure of going on the next reel of a tape file when you finish a reel of a single-file multi-reel file. On GEFRC input files it is performed automatically if you request a new read following the reading of an end-of-file mark, and for GEFRC output files it will be performed automatically when the end-of-reel marker is sensed. An end-of-file mark will be written after the last record is output and the unit switching will be performed in this case. On files defined to be multi-file multi-reel on input or output, it is the user's responsibility to perform unit switching.

2.3.6 DEFINITIONS OF GEFRC TERMS

System Input and Output Files (SYSIN and SYSOUT) - these files are designated by the file codes I* and P* respectively and are required to be in Standard System Format. They are the means by which the system reads in jobs and writes the output from all jobs concurrently in core. You may create a file in the same format as the SYSIN file but referenced by a different file code by using a $ DATA file control card with the file code desired in columns 16-17.

Standard System Format - a formate designed to allow device independence to any file assigned it. If a file is defined to be in Standard System Format, you may change the device assigned to it from tape to disc or drum, drum to disc, disc to tape, tape to printer, etc. simply by changing the device on the file control card at execution. No changes in the program coding are required. This makes it advantageous to use this format whenever possible. It is defined in the GEFRC manual under the section entitled "File and Record Format", and it can be assigned to a file by using all the "Standard Options" in the file control block macro-instruction.

Block Serial Numbers - a special word in a block to be processed by GEFRC logical record processing routines that is the first word in the block and indicates (1) the sequence number of the block in bits 0-17 and (2) the number of words in the block in bits 18-35. This word is automatically attached to the block on output files (room) must be allowed for it in the buffer and on input files it will also be assumed to be the first word of the block. This processing will be done if you specify that block serial numbers are to be processed by leaving the field in the file control block for this file null or zero.

A Logical Record - the amount of data that a program will work with at any given request. It may be a card image, a print line image, or a set of variables in a list (ie. x,y,z,xd,yd,zd, etc.), but it need not be a physical record. You may have many logical records in a given physical record. A logical record is generally termed a "record" in the GEFRC manual.

2.3.6 DEFINITIONS OF GEFRC TERMS (Cont.)

A Physical Record - the amount of data read from or written to an I/O device by any single I/O command. It would be one card on a card reader, one line on a printer, the data between two end-of-record gaps on a tape unit, etc. It is generally termed a "block" in the GEFRC manual.

Buffer - an area in your program into which a physical record is to be read or from which a physical record is to be written.

Buffer Control Word - used only when you are using a logical record processing routine of GEFRC, it is the first location in your buffer, and you must always allow room for it by adding one location to the maximum size physical record you expect on the file. Its use by GEFRC is described in the section of this report entitled "Buffers and Buffer Control Words".

Working Storage Area - an area in your program into which or from which, a logical record will be moved from or to a buffer by a logical record processing GEFRC routine. A logical record need not be the same size as a physical record.

Logical Record Type - fixed, variable, or mixed refers only to logical records and never to physical records. It defines to GEFRC's logical record processing routines, via the file control block, the method to be used in moving a logical record to or from the buffer. See the section entitled "Logical Record Formats" of this report.

2.3.7 USER'S ERROR PROCESSING ROUTINES

If, in the process of reading a data file, the system detects an error condition, it will attempt to recover a certain number of times. If after so many attempts it has failed to recover, it will assume the condition is unrecoverable and will give the operator the option of retrying the recovery procedure, calling in a user's error recovery routine, or aborting the job.

At this point if you have provided an entry in field 9 of the file control block macro-instruction and also have a routine a user's routine, the system will pass control to the location you have specified. At this point you may examine the status returned by the system or simply set a

2.3.7 USER'S ERROR PROCESSING ROUTINES (Cont.)

flag on to indicate an error condition has occurred, but under no circumstances are you to attempt any form is I/O (including printing a message). When you have completed what you wished to do for error processing, you must return control to GEFRC either by a TRA 0, 1 instruction or if you have used a SAVE pseudo-op, by means of a RETURN pseudo-op.

If you do not provide an entry in field 9 of the file control block maco-instruction or do not provide the routines referenced by that entry, and the operator requests a user's routine your program will be aborted whenever an unrecoverable error occurs on the file.

2.3.8 SYSTEM INPUT AND OUTPUT FILES

Due to the multiprogramming capabilities of the GE-635 system, it is necessary for the system to have available to it files that it can use to "stack" input and output for more than one activity at a time. These files are called "SYSIN" and "SYSOUT". Any data for your program that is a part of your input card deck and is not preceded by a $ DATA card will be put on the "SYSIN" (I*) file. If you have a $ DATA card before the data, it will be put on a file in "Standard System Format" (the same format as "SYSIN") but will be referred to by a different file code (the one in cols. 16-17 of the $ DATA card). Any file control blocks set up to accept data for either type of file ("SYSIN" or "DATA") must define the file to be in "Standard System Format".

The system output file, "SYSOUT" (P*), is also a "Standard System" file. Any data put out on this file must also contain 2 extra words per block that define the activity for which the block is being written. This 2 word logical record is inserted by GEFRC automatically, but room must be reserved for it in the buffer. The logical records within the file must contain the media and report codes in their control words. These codes are inserted automatically by calls to RPINT, PUNCH, and WTREC. Any file in the proper format may be assigned to "SYSOUT" by means of a $ SYSOUT control card.

The section entitled "System Input and Output Files" in the GEFRC manual covers these topics.

2.3.9 THE WORKING FILE CONTROL BLOCK

The working file control block is the contents of a file control block at any time during execution of your program.

It will initially contain the quantities selected by your file control block macro-instruction, but after being opened by calling the OPEN routine, it will contain a complete description of the device to be used and the location at which you are in the file at any given time.

The description of the format of this working file control block is contained in the GEFRC manual under the title "Working File Control Block Format". If your program fails during execution and you get a dump, you can always find the file control block in the dump and thereby trace where you were in the processing of that file. You also have in the file control block at this time the two "status words" returned by your last I/O request. These allow you to determine what you were attempting to do to that file just before you aborted.

2.3.10 THE FILE CONTROL BLOCK MACRO-INSTRUCTION

The file control block that you must create for each file that you wish to use GEFRC I/O routines to access is initially described by means of the file control block macro-instruction. This macro is defined in the section entitled "File Control Blocks" in your GEFRC manual.

The major consideration in deciding how to fill in the various fields of this macro is whether you are using physical or logical record processing routines to access the file. You might note by glancing at fields 3-7 that they need be filled in only if you are using logical record processing on the file.

If the file is to be in "standard system format" with the exception that it has no standard labels, then only fields 1-3 and possible 4,9, and 10 need be filled in. If the rest are left null, the standard options are chosen by default.

Field 9 entitled "error routine exit" should always be filled in for input tapes files. An address of an error processing routine in your program should be provided here.

2.3.10 THE FILE CONTROL BLOCK MACRO-INSTRUCTION (Cont.)

Note that the first field in the macro-instruction is a symbol defining the location symbol (LOCSYM) of this file control block. The address referenced by this symbol will not be the next word following the previous instruction in your program.

2.3.11 SYSTEM FILE CONTROL CARDS AND THE FILE CODE

In the portion of your input card deck following the $ EXECUTE and the $ LIMITS cards, you must provide file control cards to describe all the files (with the exception of I* and P* that you use for I/O in your program. These cards specifically describe the type of unit and its physical device number (by channel and unit) to be assigned to each file.

The key by which these cards are linked to the file control blocks in your program is the file code which is entered in columns 16-17. This is a two character alphanumeric symbol that is placed by the system in the "PAT" (Peripheral Assignment Table) along with the channel and device number and device type assigned to that symbol. This is done before your program is actually execute. At the point where you first wish to access that file, you must open the file control block associated with that unique file code by calling the GEFRC routine OPEN. Only at this point will the file control block be provided with the physical device type and address. This is done by searching the "PAT" until a match is found for the file code specified in the file control block you are opening.

Note that if a file control block is not opened, it has no reference in it to a physical device type and address. Therefore any attempt to use it to perform I/O will result in an abort.

The format of the many different types of system file control cards is described in both the GEFRC manual (in the section entitled "System File Control Cards") and in the GECOS manual (in the section entitled "Control Card Formats").

2.3.12 OPEN, CLOSE, AND THE FILE DESIGNATOR WORDS

The OPEN and CLOSE routines are used to connect and disconnect a physical device (assigned by a file control card to a given file code) to a file control block in your program that references the same file code.

Upon completing the above procedure, the OPEN and CLOSE routines will reference a set of options contained in the file designator word pointed to in their calling sequences. The bits set either on or off in that word (created by a VFD pseudo op in your program) will describe the required function that you want performed such as reqind on open, rewind on close, etc. These options and the format of the word used to describe them can be found in the section entitled "File Designator Words" in your GEFRC manual.

You must provide the address of the word containing these option flags as the first argument in the calling sequence for OPEN and CLOSE.

2.3.13 FORTRAN I/O THE $ OPTION CARD, AND THE $ FFILE CARD

When a program is written entirely in FORTRAN, there is no way by which the programmer may define his files within the program. In order to allow for as much flexibility as possible, the $ OPTION FORTRAN card asks the loader to create file control blocks for I*, P*, all files defined in $ FFILE cards, and all files for which a numeric file code is specified on a system file control card and that were not already defined on an $ FFILE card.

These files are defined to be in "Standard System Format" unless an $ FFILE card is present. If an $ FFILE card is present for a given file, the file is defined as specified in the $ FFILE card. All fields not specified on the $ FFILE card are assumed to be the "Standard Options".

As these file control blocks are created, the addresses of their LOCSYMS are stored along with the file codes associated with them in starting location is stored in location 25 octal in your program. You may find the LOCSYM of any file control block created by the loader and stored in this table by using a routine called .FGTFB and the following coding. In this example the file code is 03, and idex register two will contain the address of the LOCSYM of the file control block for file 03 when the coding is executed.

2.3.13 FORTRAN I/O THE $ OPTION CARD, AND THE $ FFILE CARD (Cont.)

```
SYMREF      .FBAD.
LDA         3,DL
STA         .FBAD.
CALL        .FGTFB
LDX2        .FBAD.
```

The $ OPTION and $ FFILE cards are defined in the GECOS manual in the section entitled "Control Card Formats".

2.3.14 THE STANDARD SYSTEM FORMAT

Due to the advantage of having device independence of files whenever possible, most third-generation computer systems have a carefully designed format that when used on a file, makes it possible to assign almost any plausible I/O device to that file without changing the program. The only change required is in the file control card assigned at the time that the program is put into execution. At this point you simply change the card from a tape card to a disc or drum or print card etc. and this allows you to use the specified device for the file.

This carefully designed format is known as the "Standard System Format", and in GEFRC you can easily define a file to consist of this format by using all of the so-called "Standard Options" in the file control block. These options are the default options (they are chosen if a field is not filled in).

This format is described in the section entitled "File and Record Format" in your GEFRC manual. Because of its adaptability, it would be to your advantage to use this format whenever possible.

2.3.15 LOGICAL VS. PHYSICAL RECORD PROCESSING

One of the many problems facing the programmer attempting to use the GEFRC I/O routines is the decision as to whether he should use a logical or a physical record processing routine for his file. The best criterion for determining which type of processing to use is simply shether or not the file has logical records blocked within each physical record. If there are no well-defined logical records, it will be wasteful both in time and space to attempt to use a logical record processing routine on the file. You would be better off simply reading or writing the whole

2.3.15 LOGICAL VS. PHYSICAL RECORD PROCESSING (Cont.)

block (physical record) into or from the area (or areas) within your program where you wish to process it.

On the other hand if the file contains well-defined logical records within each physical record, it will be a waste of time and less efficient to attempt to do the blocking/deblocking of the logical records when the system can do it automatically.

The usual reason for creating a "blocked" file is to save space on the output tape. For example: by blocking 20 logical records to one physical record instead of writing each separately as a physical record there is a savings of 19 inter-record gaps of approximately 3/4 inch with each set of 20 records.

2.3.16 LOGICAL RECORD PROCESSING, BUFFERS, AND WORKING STORAGE

A logical record is the amount of data that it is most convenient and practical for a program to handle with one request. When using one of the logical record processing routines of GEFRC, it will not always read or write a physical record on a device with each call to the routine. Instead it will cause a logical record to be accessed and, if requested, moved to (or from) a working storage area from (or to) a buffer. Only when the system decides that there are no more logical records to be accessed in a given buffer will a new request for a physical record be issued. A request for a new physical record will result in filling (or dumping) the buffer area.

The buffer area is the area set aside in the program for the maximum size physical record that will be encountered (up to 4096 words for GEFRC logical record processing routines). This buffer must have its first location reserved for the buffer control word (explained in another section of this report) and must also have room for the block serial number word if it is to be present.

The working storage area is another area of the program that must be supplied if a logical record is to be moved to or from the buffer. If you do supply its location to one of the logical record processing routines, a logical record will be moved to (or from) the buffer from (or to) this area. Only the data portion of the logical record will be moved.

2.3.16 LOGICAL RECORD PROCESSING, BUFFERS, AND WORKING STORAGE (Cont.)

Regardless of whether or not a working storage area is used, at least one buffer area must be supplied for each file using logical record processing routines.

2.3.17 BUFFERS AND BUFFER CONTROL WORDS

When processing a file with the logical record processing routines of GEFRC, you must define at least one buffer area within your program and supply its address to the file control block that will be used on that file. This area must be large enough to contain the largest physical record expected on this file plus one extra word for the buffer control word that GEFRC will use in processing the data. This buffer size is limited to 4096 words for GEFRC logical record processing.

The function of the buffer control word is to control the data flow to or from the buffer during the time the physical device is being accessed, but it is also used during the processing of logical records in the buffer. For input files it will contain the address of the last data word plus one of the current logical record in bits 0-17. For output files it will contain the number of words of the buffer that have been filled up to this point in bits 0-17.

Note that due to the function of this word, the same buffer for both input and output should not be used unless you can be sure that all reading will be done before you attempt to use it for writing or vice versa.

This subject is covered in the GEFRC Manual CPB-1003 in the section entitled "Buffers".

2.3.18 LOGICAL RECORD FORMATS

When using the logical record processing routines of GEFRC, it is necessary to define (by means of the file control block macro-instruction) the format of the logical records of the file. GEFRC allows the definition of these logical records to be variable length, fixed length, or mixed length. It must be decided which of these types best fits the format of the file that is to be processed. The section entitled "File and Record Format" in the GEFRC manual gives

2.3.18 LOGICAL RECORD FORMATS

a brief description of each of these formats. The following paragraphs supplement those descriptions.

When GEFRC is processing a file defined to contain "variable length" logical records, it will expect bit 0-17 of the first word of each logical record to contain the record size. If this is not true of the data that is to be processed, GEFRC will still attempt to use bits 0-17 of the first word to decide how many words to process. This can only result in a complete failure in processing.

When GEFRC is processing a file defined to contain "fixed length" logical records, it will assume that every record conforms to that specified record size. It will pick off that many words and move them to or form the buffer regardless of how the data is actually arranged. If the data format does not fit this length, the program will lose data or get out of sequence after the first record access.

When GEFRC is processing a file defined to contain "mixed length" logical records it requires the following information.

> For input files, field 7 of the file control block macro-instruction must contain the address of a routine that will determine the record size and store it into word 1 (LOCSYM+1) of the file control block. This routine will be called automatically from GEFRC whenever a logical record is requested.
>
> For output files, the logical record size must be stored into word 1 (LOCSYM+1) of the file control before attempting to put out a logical record.

2.3.19 PHYSICAL RECORD PROCESSING AND THE DATA CONTROL WORD

When the input or output format does not consist of well-defined logical records, it is an advantage to use the physical record processing routines of GEFRC (READ and WRITE). These routines also allow you to input or output Physical Records of greater than 4096 words (the limit of a Physical Record for all Logical Record Processing Routines).

C.2

2.3.19 PHYSICAL RECORD PROCESSING AND THE DATA CONTROL WORD (Cont.)

When using either READ or WRITE routines, one or more data control words must be specified in order to define (1) the starting address of the area you wish to read into or write from, (2) the number of consecutive words you wish to read or write, and (3) if the field is assigned to random Disc or Drum, the first word of the sequence msut contain the relative block address of the block to be read or written.

There are four possible actions that any of the data control words of a sequence may perform. The action desired must be specified by coding one of the four pseudo-operations in the operation field.

Pseudo-op	Explanation	Binary Action Code
IOTD	Transmit and Disconnect-Move the number of words specified by the count to or from memory and then terminate the I/O.	00
IOTP	Transmit and proceed-Move the number of words specified by the count to or from memory and then continue on to the next data control word and do as it indicates.	01
TDCW	Transfer to DCW-Don't move any data Go to the DCW at the address specified in this DCW and do as it indicates.	10
IONTP	Nontransmit and proceed-Skip the number of words specified in the count of this DCW, and then go on to the next sequential DCW and do as it indicates.	11

The following is an example of reading variables off block 10 of random drum storage. The variables are X,Y,Z,X,Y,Z but whereas X, Y, and Z are consecutive in the block and X, Y, and Z are consecutive, there are 20 other variables between Z and X that you don't care about. At another point in your program, you have already set up DCW's to read X, Y, Z; therefore you need only use a TDCW pseudo-op to make use of those DCW's already created. Needless to say that is a hypothetical case and not very probable.

2.3.19 PHYSICAL RECORD PROCESSING AND THE DATA CONTROL WORD (Cont.)

```
        CALL     READ  (FCB, DCWIN)   *Read in Variables
         .                 .                 .
         .                 .                 .
         .                 .                 .
DCWIN    DEC      10                   *Block address on drum
IOTP     IOTP     X,1                  *Read in data
         IOTP     Y,1                  *
         IOTP     Z,1                  *
         IONTP    Z,20                 *Skip 20 words
         TDCW     NXTDCW               *Go to the next set
         .                 .                 .
         .                 .                 .
         .                 .                 .
NXTDCW   IOTP     XD,1                 *Read reset of data
         IOTP     YD,1                 *
         IOTP     ZD,1                 *Stop I/O after this
```

The following is the format in memory of a data control word entitled "use" of IOS". The GEFRC manual covers these words lightly under the descriptions of the read and write routines.

0	17 18	21-22-23-24	35
Data address	MDZ	Action code	Word count

The GECOS Manual CPB-1518 has information on these "DCW" words in the section entitled "Use of IOS". The GEFRC Manual, CPB-1003 also covers ther words under the descriptions of the READ and WRITE Routines.

2.3.20 GE-635 L* LIBRARY SUB-ROUTINES

The following is a list of subroutines listed on the GE-635 L* Library. This listing is provided for general reference of the reader. It should be noted that the sub-routine listed here are normally used by the Systems Software. Programmers should not attempt to call any of these routines directly without careful review with Systems personnel of the calling procedures.

FCXP - Exponential - Complex Base 8 Integer Exponent
FCSN - Complex sine and cosine
FCSQ - Complex Square Root
FCLG - Complex logorithm
FCEX - Complex exponential
FCAB - Complex absoulute Value
FCMP - Complex multipication and division
FDXP - Double precision Exponential
FDAT - Double Precision Arc Tangent
FDSN - Double Precision Sine and Cosine
FDSQ - Double Precision Square Root
FDLG - Double Precision Logorithm
FDEX - Double precision Exponential
FDMD - Double Precision Modulus
F1XP - Exponential - Integer Base and Exponent
F2XP - Exponential - Floating Point Base Eight Exponent
F3XP - Exponential - Real Base and Exponent
FTNH - Real Hyperbolic Tangent Base and Exponent
FATN - Real Arctangent
FSIN - Real Sine and Cosine
FSQR - Real Square Root
FALG - Real Logarithm
FEXP - Real natural exponential
FSLO - Short List I/O interface
FRDB - Binary I/O interface
FVFI - Name list input
FSTU - Pre-execution initializer
FDBG -
FBLO - Short list binary I/O interface
FRDD - BCD I/O interface by format Control
FINC -
FBST - Back space record
FEOF - End file processor
FFEE - Initialization of end of file processing
FDMP - Memory dump
FVFO - Name list and debug output
FSLW - Carriage control simulator
FEFT - Rewind and end of file processor

2.3.20 GE-635 L* LIBRARY SUB-ROUTINES (Cont.)

FLIT - Sense light simulator
FSWI - Sense switch test
FSET - Define logical file table
FFER - Initialization of data error processing
FCLO - File closing
FOPE - File opening
FXEM - Execution error monitor
FDPT - Double Precision powers of ten
FSLI - Short list I/O processor
FDCK - Exponent over flow and divide check test
FLVK - Restores link during execution
FSTO - Access half of a double precision or complex word
FFLT - Fault processor
FLHS - Restore main link and FCB from H*
CMAA - Working storage and dating
CMAB - Program linkage control
CMAC - Linkage control stock
CMAD - Display - 2 comparison (Commercial collating SE)
CMAE - Characters set changer
CMAF - Checkpoint memory dump
CMAZ - COBOL set-up subroutine
CSAG - Sort/Merge interface
CGAH - Open file interface
CGAI - Report line / .CBNTR interface
COAJ - Report line output
COAK - Gather write
COAL - Display upon typewriter
COAM - Display upon sysout
CIAN - Scatter read
CIAO - Accept from console and typewriter
CIAP - Accept from GEIN
COAQ - I/O Buffers
CNAR - BCD to binary floating
CNAS - Binary to BCD floating
CNAT - BCD to binary fixed
CNAU - General exponentiation
CEAV - Control Character move
CEAW - Control - Character Move, BWZ
CEAY - Control - Character Move, Signed, BWZ
CEAX - Control - Character Move, Signed
CEAZ - Control - Zero Replacement
CEBA - Control - Zero Replacement, BWZ
CEBB - Control - Zero Replacement, Signed
CEBC - Control - Zero Replacement, Signed, BWZ
CEBD - Control - Floating Zero Replacement
CEBE - Control - Floating Zero Replacement, BWZ
CEBF - Control - Floating Zero Replacement, Signed
CEBG - Control - Floating Zero Replacement, Signed

2.3.20 GE-635 L* LIBRARY SUB-ROUTINES (Cont.)

CEBH - Control - Literal Character Move
CEBI - Control - Literal Character Move, BWZ
CEBJ - Control - Literal Character Move, Signed
CEBK - Control - Literal Character Move, Signed, BWZ
CEBL - Control - Literal Zero RPL.
CEBM - Control - Literal Zero RPL., BWZ
CEBN - Control - Literal Zero RPL., Signed
CEBO - Control - Literal Zero RPL., Signed BWZ
CEBP - Control - Literal Float Zero RPL.
CEBQ - Control - Literal Float Zero RPL., BWZ
CEBR - Control - Literal Float Zero RPL., Signed, BW
CEBT - Control - Justified Character Move
CEBU - Control - Justified Character Move, BWZ
CEBV - Control - Justified Character Move, Signed
CEBW - Control - Justified Character Move, Signed B
CEBX - Control - Justified Zero RPL.
CEBY - Control - Justified Zero RPL., BWZ
CEBZ - Control - Justified Zero RPL., Signed
CECA - Control - Justified Zero RPL., Signed BWZ
CECB - Control - Justified Float Zero RPL.
CECC - Control - Justified Float Zero RPL., BWZ
CECD - Control - Justified Float Zero RPL., Signed
CECE - Control - Justified Float Zero RPL., Signed
CECF - Control - Justified Float Zero RPL., Signed
CECG - Control - BIN/BCD Move, BWZ
CECH - Control - BIN/BCD Move, Signed
CECI - Control - BIN/BCD Zero RPL.
CECK - Control - BIN/BCD Zero RPL., BWZ
CECL - Control - BIN/BCD Zero RPL., Signed
CECM - Control - BIN/BCD Zero RPL., Signed, BWZ
CECN - Control - BIN/BCD Float Zero RPL.
CECO - Control - BIN/BCD Float Zero RPL., BWZ
CECP - Control - BIN/BCD Float Zero RPL., Signed
CECQ - Control - BIN/BCD Float Zero RPL., Signed, BW
CECR - Control - BIN/BCD Justified Move
CECS - Control - BIN/BCD Justified Move, BWZ
CECT - Control - BIN/BCD Justified Move, Signed
CECU - Control - BIN/BCD Justified Move, Signed
CECV - Control - BIN/BCD Justified Zero RPL.
CECW - Control - BIN/BCD Justified Zero RPL., BWZ
CECX - Control - BIN/BCD Justified Zero RPL., Signed
CECY - Control - BIN/BCD Justified Zero RPL., Signed, BW
CECZ - Control - BIN/BCD Justified Float Zero RPL.
CEDA - Control - BIN/BCD Justified Float Zero RPL., BWZ
CEDB - Control - BIN/BCD Justified Float Zero RPL., Sign
CEDC - Control - BIN/BCD Justified Float Zero RPL., Sign
CEDD - Operational - Character Move
CEDE - Operational - Zero Replacement

2.3.20 GE-635 L* LIRBRARY SUB-ROUTINES (Cont.)

CEDF - Operational - Floating Zero Replacement
CEDG - Operational - Literal Character Move
CEDH - Operational - Literal Zero Replacement
CEDI - Operational - Literal Floating Zero Replacement
CEDJ - Operational - Justified Character Move
CEDK - Operational - Justified Zero Replacement
CEDL - Operational - Justified Floating Zero Replace
CEDM - Operational - Binary to BCD Fixed Conversion
CEDN - Operational - Blank when Zero
CEDO - Operational - Sending Sign De-Edit
CEDP - Operational - Receiving Sign Edit
CEDQ - Operational - Property Vector Word Decode
CEDR - Operational - Set Optional Editing
CEDS - Operational - Move/Edit Constants and Save
CMDT - Operational - Correction Linkage Stopper
CCDX -
GBSR - Backspace in records
GFSR - Forward in records
GBSF - Backspace tape in files
GFSF - Space forward in files
GREW - Rewind unlabeled tape/linked Disc Drum
GRED - Physical Record Read
GWRT - Physical Record Write
GWAI - Wait For Read/Write Complete
GWEF - Write EOF
GSTI - Make A File An Input File
GSTO - Make A File An Output File
GEPR - EDIT A Line Before Printing
GPRT - Entry point for print
GPUN - Entry point to the punch
GRDR - Read Record
GALT - Alter function of RDREC
GRDC - Read Card
GWRC - Places record in Specified file
GIOP - Page Number
GLCD - Card Labels
GIOE - Entry for I/O Edit
GGTB - Buffered Input
GPTB - Buffered Output
GPSZ - Sets Actual Record Size
GFCE - Force End of reel
GOPE - Opens Multiple files
GCLO - Close multiple files
GREL - Release functions
GBCD - BCD to binary
GBBC - Binary to BCD
G20R - Common Write
G25E - Common Read

2.3.20 GE-635 L* LIBRARY SUB-ROUTINES (Cont.)

G50R - Analyse FCB Exceptions
G27R - Analyse I/O Status Return
G37R - Call Users error routine
G60R - Write MSG. Before abort
G80R - Create Select Sequence
G90R - Issue Select
GLAB - Call GFRC Overlay
GINI - Create Output Labels
GRCV - Tape Error Recovery - When Block Ser. # Differ Sort

2.3.21 GE-635 *L SUB-ROUTINES

The following is a list of local user routines available on the GE-635 *L Library:

BESFIT - Determines which degree curve best fits the calibration data.
CURFIT - Determines the coefficients of calibration curve from the calibration data.
INVMAT - Matrix Inversion routine.
ISARG - Used with CURFIT to determine the number of arguments.
CKID - Checks program ID.
DBUG1 - Provides a standardized message on output data indicating the program used is not a production deck.
DMP - Tape Dump.
BDC - Binary to Decimal conversion.
BHC - Binary to Hollerith conversion.
ARSIN - ARCSIN function.
ARCOS - ARCCOS function.
DARSIN - Double precision ARCSIN function.
HALT1 - Stops program prior to on-line printing to allow paper change.
GOPNA - Force input files to unlabelled.
GOPNB - Force all files to unlabelled.
DATER - Returns date in form "MMDDYY".
EDATE - Convert "MMDDYY" to "MM/DD/YY".
MODATE - Convert "MMDDYY" to MON,DD,19YY".
YRDAY - Convert "MMDDYY" to OYYDDD".

2.3.21 GE-635 *L SUB-ROUTINES (Cont.)

For Use With COBOL Programs

JUDAT
MADT
MARD
MAFUL
CONDSE
ABENDU
ASJUDA
QZDT
QZRD
QZFUL
PARMIN
PHO5RS

4020 Plotter Sub-Routines

Details for these routines can be found in SC-4020 Computer Recorder DOC. No. 9500056.

CAMRAV	POINTV	RETE2V
GRID1V	APLOTV	VCHARV
LINRV	APRNTV	TABL1V
MGRIDV	PLOTV	TABL2V
MLINV	IXV	TABL3V
DXDYV	IYV	TABL5V
PLOTFT	NYV	TABL6V
CLEAN	NXV	TABL7V
PLCNTV	UYV	TABL8V
XSCALV	UXV	TABL9V
YSCALV	SERSAV	TABL11V
LABLV	SCLSAV	TABL15V
BNBCDV	SETMIV	
RESETV	CHARV	
FRAMEV	IDV	
LINEV	FORMV	
PRINTV	BUTTV	
BRITEV	NORFV	
FAINTV	GYAXIV	
STOPTV	CHSETV	
SETCOV	BUFV	
SETCIV	NWDSV	
SETMOV	PLOUTV	
HOLDOV	CHSIZV	
HOLD1V	RITSTV	

Descriptions of these sub-routines are available in NASA Systems Programming Section, IN-DAT-4.

2.3.22 COMPUTER PROGRAM DOCUMENTATION

It is mandatory that all computer programs be documented in a prescribed format prior to their release into production. This format is detailed in "STANDARD FOR PREPARATION OF COMPUTER PROGRAM DOCUMENTATION", dated March 11, 1971. This document is available to the programmer through his immediate supervisor.

2.3.23 ABORT/DELETE REASON CODES

These codes will appear in the Loadmap portion of the listing directly below the "BEGIN ACTIVITY" message in the event of an abnormal termination of the job.

Abort terminations can be categorized into three main groups. Abort during execution, abort, abort during I/O and abort while loading. Information on these categories can be seen in the following manuals.

CPB-1518	(GECOS III)	Abort during execution.
CPB-1003	(GEFRC)	Abort during I/O.
CPB-1008	(GENERAL LOADER)	Abort while loading.

The following is a list of the abort codes and a brief explanation of each.

CODE	EXPLANATION
0	Excessive output of GESYOT.
01	Improper GESYOT call sequence code.
A	More than 638 links requested on $ DISC/$ DRUM linked file.
A1	Peripheral Assignment Table (PAT) buffer size exceeded.
A2	Allocation impossible because of channel requirements (devices requested are in excess of those available on a given channel).
A3	Secondary tape buffer size exceeded.
A4	Peripheral requirements in excess of actual resources have presented an impossible allocation task.
A5	Abeyance table size exceeded.
A6	More than 6 output files requested.
A7	Identical logical unit designators have been used for mass storage requests on the same type of device.

2.3.23 ABORT/DELETE REASON CODES (Cont.)

CODE	EXPLANATION
A8	Allocation impossible because of channel requirements (as previously defined by one or more peripherals in abeyance).
A9	Named device not configured, "RLSE'd," or no such name.
AD	Zero links requested on $ DISC/$ DRUM card of improper deck setup.
B1	Variable error -- Improper use of variables on a $ OUTPUT or $ INPUT card. Check for spelling, use of MIXED for other than card input, use of MIXL on input, etc.
B2	Variables on the $ MULTI control card are incorrectly used.
B3	Improper request for transliteration. Either the media combination is in error or codes (IBMF, GE225, etc.) are improperly interpreted. Transliteration applies only to the reader; assigning it to any other device will cause a B3 abort.
B4	Input or outpu device code is not acceptable to BMC.
B5	A partial header label has been encountered on magnetic tape input.
B6	User has not provided an Sxxxxx or Uxxxxx option (input only), and an error has occurred.
B7	The number (xxxxx) of acceptable input errors as specified in Uxxxxx or Sxxxxx has been exceeded.
B8	Input error prevents BMC from continuing.
B9	An entry made to an overlay is erroneous; probable hardware failure.
C0	A requested input/output file is not contained in the file address table; an internal problem in the compiler.
C1	Drum address table (*3 File) is full. Normally, this problem may be corrected by increasing the size of the *3 File with a $ DISC or $ DRUM card. Maximum limits for the *3 File is 50 random links.
C2	Fixed portion of the data name table is filled to the point that the largest record (01) contained on the overflow file cannot be read back into memory. Normally, this problem may be corrected by including a $ LIMITS card to extend memory limits, thus enabling extension of the data name table.
C3	Invalid internal list structure of the Data Division, possibly caused by invalid level structures for some data items which have been processed prior to this point.

2.3.23 ABORT/DELETE REASON CODES (Cont.)

CODE	EXPLANATION
C4	Substantive Stack build error in Report Writer. Occurs if the substantive stack is empty and the overflow indicator is not on when the Substantive Stack Popup routine is called.
C5	Invalid internal list structures of the Report Writer, which is an internal Compiler problem.
C6	Fixed portion of the Report Table is full.
C7	Fixed and variable portion of the Report Table is full.
C8	Invalid internal list structures of the Report Writer.Analyzers, which is an internal Compiler problem.
C9	Report Writer Generator error; Report Generator is unable to build its COBOL lists (internal language) using the Analyzer build routines. An internal Compiler problem.
CA	Invalid internal list structures of the Report Writer Generators, which is an internal Compiler problem.
CB	Invalid (not 17_8 or 23_8) end-of-file mark has been encountered in a COBOL or FORTRAN object program.
D1	IDS has aborted program for cause, as indicated by its abort message.
D2	User error.
EF	Input tape in improper format for PRODUCT.
E(n)	Undefined GEPR override option reuqested. (n) = absolute value of first status word when the I/O request was initiated.
EP	Irrecoverable I/O Error. (IC will contain location of status return pointer; indicators will contain courtesy call address, if any.)
EX	EXECUTE cutton at processor T & O Panel depressed (Operator-initiated abort). Type-out will appear at the console.
F0	Memory address fault (attempt to access memory beyond BAR).
F1	Fault tag (tag field -40).
F2	Command fault (slave execute of CIOC, RMCM, SMCM, RMFP, SMFP, SMIC).
F3	Derail fault (Op code = 002).

2.3.23 ABORT/DELETE REASON CODES (Cont.)

F4	Lockup fault (the processor is in a program lockup which inhibits recognizing an Execute Interrupt or Interrupt-type fault for greater than 16 milliseconds).
F5	Connect fault (Control processor, in master mode, executed CIOC addressed to a memory port containing a processor (the latter processor incurs the connect fault).
F6	Parity fault (A parity error was detected in a word which was read from a core location.
F7	Zero Op Code fault
F8	Op Not Complete fault
F9	Overflow/Underflow fault
GF	GEFRC has aborted program for cause as indicated by its abort message.
G1	Input Header Label
G2	Error in Input Tape Header label on tape
G3	Error in Output Tape Header label
G4	Error in old label on output tape
G5	End-of-tape while writing header label
G6	Blank tape rather than input trailer label
G7	Block count error in input tape trailer label
G8	End-of-tape detected on multifile reel. (Note: A multifile reel cannot be continued on a second tape).
G9	New FCB address is the same as present one in the MME GEFCON sequence.
I0	Divide Check fault
I1	Invalid address in MME
I2	Improper use of MME during courtesy call
I3	File code not in Peripheral Assignment Table (PAT), or user attempts to execute a MME GEINOS with a file code specifying a SYSOUT file.
I4	Invalid PAT pointer
I5	No file pointer in GEINOS select sequence
I6	GEENDC used when not in courtesy call
I7	Attempt to access outside file limits
I8	Allowable run time exceeded
I9	Too long in courtesy call
J0	No activity defined. Will occur only in the last activity of a job.
J1	$ PRODUCT card does not contain a program name

2.3.23 ABORT/DELETE REASON CODES (Cont.)

CODE	EXPLANATION
J2	$ DATA card does not contain a file code
J3	$ INCODE request improper
J4	$ SYSOUT card does not contain a file code
J5	Invalid option of a $ system call card ($ GMAP, $ IDS, etc.)
J6	Invalid $ file card.
J7	Maximum number of files exceeded
J8	Invalid $ NTAPE card
J9	Input read error
JA	Job has exceeded Input Media (GEIN) storage size or: Job has been aborted through no fault of its own so that links may be released and system hang-up prevented. Resubmit job.
JB	Checksum error on binary card
JC	More than one IDS file per activity
JD	$ PERM name not found in catalog
JE	Check character alert on system storage file
JF	Illegal device type found in catalog
JG	Catalog logical device designator does not match that of configuration table.
JH	System card reader inoperable (card jam, etc.)
JI	9SA not configured.
JJ	A $ PPT card encountered in input deck; $ PPTR or $ PPTP must be used.
JK	Job has zero storage request on $ LIMITS card.
JL	A $ REMOTE card has been encountered by GEIN but the DATANET-30 processor not configured.
JM	Invalid data on permanent file.
JN	A $ SELECT card has been used within a Select activity. "Nesting" of $ SELECT cards is illegal.
K1	Improper device code in PAT or CONFIG table
K2	Invalid status return point for I/O
K3	Invalid file pointer for I/O
K4	Invalid DCW pointer for I/O
K5	Invalid courtesy call address
K6	Invalid Op code, linked file I/O
K7	Two successive TDCW's
L1	Missing subroutine required
L2	NOGO option exercised or Program and Loader overlap.
L3	Fatal error encountered during loading

2.3.23 ABORT/DELETE REASON CODES (Cont.)

CODE	EXPLANATION
M0	Parameter A in GETMORE calling sequence is not 0, 1, 2, or 3.
M1	Illegal device type in GECALL/GERSTR.
M2	Entry not present in GECALL/GERSTR disc catalog
M3	Entry not present in GECALL/GERSTR tape catalog
M4	Memory limit exceeded in GECALL/GERSTR
M5	File code missing in PAT for GECALL/GERSTR
M6	Input error in reading within GECALL/GERSTR
M7	Parameter A in the GEMORE calling sequence specifies additional links but the Slave PAT specifies a nondisc or nondrum device or: GEMORE request for additional links results in a total exceeding 638 links.
M8	Production Library file has not been configured at Startup.
M9	Calling sequence to GESYOT destroyed between the MME GESYOT and processing of the MME.
MA	Calling sequence to MME GERELS contains (1) zero files to be released, (2) 20 or more files to be released, (3) program number 73_8 or (4) file word contains characters other than two low-order characters in slave mode.
N1	Fault Tag used
N2	File code not in Peripheral Assignment Table (PAT) for MME GEFILS.
N3	Illegal action during rollback.
N4	Request for GESAVE exceeds memory limits or is zero.
N5	Illegal device or no PAT for GESAVE
N6	Catalog full in GESAVE
N7	I/O error in GESAVE
N8	Insufficient number of links for GESAVE.
N9	Too many DCW's required for GESAVE.
P0	Maximum of 63 levels of MACRO expansion exceeded
01	Logical Unit Table overflow
02	Missing Logical Unit Table
03	No space for Logical Unit 6 Buffer
04	Machine error or unexpected error to FORTRAN Compiler

2.3.23 ABORT/DELETE REASON CODES (Cont.)

R1	Fatal ALGOL error. An error number preceding the R1 designation categorizes the error.
R2	Unsuccessful attempt to write data to the DATANET-30 in the direct access mode - MME GEROUT
R3	Remote opertor abort (GERTS)
R4	GERTS too busy to accept any new requests; resubmit job.
R5	Illegal op code for MME GEROUT.
R6	Illegal file code in MME GEROUT.
R7	Invalid media code for op code 1 - MME GEROUT.
R8	Invalid media code for op code 7 - MME GEROUT.
R9	Unsuccessful attempt to write data to disc - op code 1 or - MME GEROUT.
RA	Excessive word count in direct access data block.
RB	Excessive character count in direct access data block.
RC	GERTS is not on the system; DATANET-30 is not configured.
RD	Excessive word count for data in op code 1 or 7 of MME GEROUT.
RE	Excessive output on remote file.
RF	Illegal address in MME GEROUT calling sequence or associated data block.
RG	Attempt to write to an unopened file - MME GEROUT.
SC	SORT abort; reason indicated on execution report.
SE	System Editor has encountered an irrecoverable error.
SM	COBOL Linkage not in stack
U1	Control deck error
U2	Comparison error
U3	Hardware error
V1	Compiler error abort. The JOVIAL statement is given in a corresponding error message on the listing.
V2	Object program short. The JOVIAL Compiler generates a MME GEBORT when replacing an erroneous JOBICAL statement.
X1	Operator deleted job from control stack.
X2	Operator aborted job in execution.

SECTION 3 TESTING AND DEBUGGING AIDES

3.1 PROGRAMS

3.1.1 CTEA COMPRESSED TAPE ERROR ANALYSIS

This program examines the ACE 606 compressed data tape for discrepancies in format, time spikes, and time jumps.

The following messages are output along with a dump of the record:

```
*  ILLEGAL PRIME FRAM NUMBERS                                  RECORD XXXX
   PF MARK MISSING AFTER STAGING MARK                          RECORD XXXX
   PF MARK MISSING AFTER LOS MARK                              RECORD XXXX
*  DOUBLE INITIALIZATION MARKS                                 RECORD XXXX
*  LOS AT XXX DAYS XX HRS XX MIN XX.XXX SEC                    RECORD XXXX
*  STAGING AT XXX DAYS XX HRS XX                               RECORD XXXX
   MIN XX.XXX SEC
   ODD NUMBER OF DATA-TAG BYTES                                RECORD XXXX
   STAGING ZONE MISSING FROM FIRST RECORD                      RECORD XXXX
   PF NOS. NOT SEQUENTIAL IN INITILIZATION                     RECORD XXXX
*  PARITY ERROR                                                RECORD XXXX
*  TIME SPIKE     HR      MIN      SEC          RECORD
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
*  TIME JUMP      HR      MIN      SEC          RECORD
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
*  TIME CHANGE    HR      MIN      SEC          RECORD
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
                  XX      XX       XX.XXX       XXXX
```

* A dump of the record will not follow these messages.

3.1.2 CDGD ACE COMPRESSED DATA DUMP.

This program can be used to dump a tape in the ACE Compressed Data Tape Format. Dump options are: Straight octal dump, analytic dump, and tag dump. Output will be either a SYSOUT tab or a BCD tape which can be dumped by a BMC routine. The tape output option should be used if the output is expected to exceed 10,000 lines.

3.1.2 CDGD ACE COMPRESSED DATA DUMP (Cont.)

ABSTRACT

This program can be used to dump tapes in the ACE Compressed Data tape Format. There are three (3) types of dumps that can be obtained, and several options for each type. A brief discussion of the three (3) dump types is given below, with a more detailed discussion in Section V:

Straight Octal Dump:

This dump in byte pair (byte=an entry on the CDT) all entries on the input tape. Loss of syncs, stagings, and parity errors will be noted. The user must decode this dump for his own use.

Analytic Dump:

This dump is broken into two (2) parts:

Straight octal dump of the first ten (10) records.
Analysis dump of remaining records. The analysis indicates:
Record numbers
Loss of sync
Staging
Parity
Blank records
End of File

Tag Dump:

This dumps out only the tags, and associated data, that are specified by the requestor. These tags may be input either from cards or a Check Tape. Other information output for identifying these data-tag pairs is:
Record number
Prime number
Loss of sync
Staging
Prime fram 1, LOS and Staging time

As an additional option, the data (not tags) can be output in either the octal or decimal number system.

Output of this program is either a printer output or a BCD tape which can be dumped by a BMC routine.

3.1.2 CDGD ACE COMPRESSED DATA DUMP (Cont.)

Program Output Format

Tape:

This option creates a BCD tape which can be printed with a BMC dump program. This output format is identical to the output format for the SYSOUT output option.

Printer (SYSOUT):

The output format depends on the Dump Option selected:
Straight Octal Dump:
Byte pairs (data-tag) from the input tape are grouped in the following manner (D=data, T=tag)

```
DDDDTTTT    DDDDTTTT    DDDDTTTT  etc.
```

Analytic Dump:

The first ten (10) records are dumped as in the Straight Octal Dump Option. The remaining records are processed with only the following conditions being noted: Record number, Loss of Sync, Staging, Parity errors, Blank records, and End of Files.

Tag Dump:

Only the data associated with specified tags is dump. All conditions which are noted by the Tag Dump also.

A summary is printed at the end of each dump, indicating totals for the following:
Number of Stagings
Number of Loss of Syncs
Number of Parity Errors

Number of Blank Records

3.1.3 CTAC-COMPRESSED TAPE ANALYSIS COPY

This program will provide an error analysis check of the raw ACE tape format and output initialization times. Prime frame one times will be checked for continuity. There are four options. The program can provide an identical copy of the tape. The program can copy a specified span of the input tape (this copied span can be read directly into PACD without operator intervention when PACD is batched with

3.1.3 CTAC-COMPRESSED TAPE ANALYSIS COPY (Cont.)

this program). The program can output a total data-tag dump between specified limits. Finally, data for quality assurance (Q.A.) can be output in an easy-to-use format between specified limits. Limits for the options may be specified either by time or record number.

3.1.4 SCTP STRAIGHT CHANNEL TAPE PRINT

Prints out specified data words of variable size from magnetic tape, with associated time, and the update status of the data word.

To be used with Mission output the FR1400 tape output. For additional information check program document SCTP #4001.

3.1.5 TDTD TELEMETRY DATA TAPE DUMP

Prints out data in a prescribed format from a telemetry data tape. To be used with Mission output and FR1400 tape output. For additional information check program document TDTD #1045.

3.1.6 TTGP TEST TAPE GENERATION PROGRAM

This program is available for programmers to use in generating test tapes for program checkout. The program (TTGP1) accepts as input a telemetry tape and generates a shorter test tape or tapes in a similar format, the program has the capability of modifying the time word at the user's option to simulate any of the following time conditions:

Time Spike
Time Recycle
Time Drop-Out

This will enable the user to test his time edit routine. Time may be in TEL2 or DATA-CORE format. The program has the option of modifying data words to insert sync values and generating multi-reel output.

3.1.7 $ UTILITY, $ FUTIL

The following options are available: COPY, COMPARE, SKIP, DUM, REWIND, HOLD.

Reference the GECOS manual for detailed information.

3.1.8 1005 DUMPS

If files are skipped on the 1005 and printing is started on the last group of records, the record count will be accurate because the 1005 continues to count the records even though the records are not listed.

SECTION 3 TESTING AND DEBUGGING AIDES

3.2 SUB-ROUTINES

3.2.1 CURVE FIT BY LEAST SQUARES METHOD

Purpose

To compute the least squares estimated coefficient for a given set of data.

Usage

Calling Sequence - CALL CURFIT (X,Y,N,L,A)

Where X = Independent Variable vector (floating point, double precision)

Y = Dependent Variable vector (floating point, double precision)

N = Number of points (fixed point)

L = Number of coefficients; i.e., degree + 1 (fixed point)

A = Coefficients vector (floating point, double precision)

CURFIT uses 2901 words.

Error Condition

A message indicates when matrix inversion for computation of coefficients cannot be performed. The coefficients will be computed but will not be valid.

Restrictions

Data to be fitted must be in double precision.
Maximum limits are 101 points and 10th degree equation.

3.2.2 CKID CHECK ID

Purpose

To insure that the input data deck matches the program deck by ID number.

3.2.2 CKID CHECK ID (Cont.)

Usage

Calling Sequences - CALL CKID (A) CALL CKID (A,B,)
Where A = Location of five-character program ID, left justified.*
B = Location of the File Control Block to be used to read in the ID card.
CKID uses 517 words.
No error conditions.

*If the word DEBUG is used in this location it will call up the DBUG1 subroutine.

Restrictions

None

Method

Reads in ID card, prints columns 6 through 80 and compares the card ID, columns 1 through 5, to the internally starred program ID.

If the card ID and the program ID do not match, the following message is written on SYSOUT and the program is aborted with a CK abort code. (MME GEBORT).
PROGRAM ID (XXXXX) DOES NOT CHECK WITH CARD ID (XXXXX).

If only one argument is used, the subroutine checks to see whether a $ OPTION FORTRAN (or FCB) control card was used (cell 25_8 1 0). If so, the location of CKID's FCB for I* is started in the FORTRAN FCB list. Then, in either case, CKID's I* FCB is opened and GET is used to read the ID card.

The ID information is written on SYSOUT using CKID's P* (Execution report code) using IOEDIT and PRINT. The location of this FCB is not stored in the FORTRAN FCB list.

3.2.3 DMP TAPE DUMP

Purpose

To provide the capability of dumping any span of any file of a GE-635 magnetic tape in either floating point, integer or octal notation.

3.2.3 DMP TAPE DUMP (Cont.)

Usage

Calling Sequence - CALL DMP (T,P,F,M,S,N,)

Where T = Location of file control block of tape to be dumped.

P = Location of file control block of print file.

F = File Number to be dumped.

M = Dump mode.
0 = Integer
1 = Octal
2 = Floating Point

S = Number of record at which dump is to start.

N = Number of records to be dumped.

DMP uses 5376 words.

Error Conditions

3.2.4 DBUG1 Debug SUBROUTINE

ABSTRACT:

There are occasions that require the processing of data by a program before the program has been completely checked out and placed in production. These occasions occur when test data is not available prior to a test to checkout a program, or when a program has been placed in production and a bug has been detected and a correction made. When this occurs, programming personnel normally run the program. It has been necessary in the past that data resulting from these runs be marked as "test data" before being issued. The DBUG1 subroutine provides an automatic means of having the "test data" message printed on the output, and should become a standard feature for all programs producing printed data.

FUNCTION:

When called by the appropriate calling sequence, DBUG1 examines the CKID test word for a "D" in the least significant character. If the character "D" is present, the subsoutine then moves a 132 character print line containing

3.2.4 DBUG1 DeBug SUBROUTINE (Cont.)

FUNCTION: (Cont.)

the appropriate print message to the 22 computer word print line indicated by the second argument of the calling sequence. If the last significant character of the CKID test word is not a "D" (octal 24), then 132 characters of blanks are stored in the designated print line. The operation of DBUG1 is dependent upon the CKID subroutine, which is also a programming standard.

The CKID subroutine, when called, will examine the program ID, as found on the CKID card, and compare it with the contents of the CKID test word specified in the calling sequence, as it normally does. If the compare is successful, CKID will also test columns 6-10 of the CKID card for the word DEBUG. If present, CKID will place a "D" (octal 24) in the least significant character of the CKID test word. It is this information for which the DBUG1 subroutine tests of determine whether to transfer the "test data" message or to transfer blanks.

USAGE:

To use DBUG1, columns 6-10 of the CKID card must contain DEBUG if the test data message is to be printed. If anything other than these characters appear in columns 6-10, then no test message will appear.

Calling sequence:
CALL DBUG1 (A,B)

where:

A = location of CKID test word
B = location of first word of a 22 computer word print line to which the Test Data message is to be transferred if required.

DBUG1 requires 60 (decimal) words memory.
NOTE: IT IS THE RESPONSIBILITY OF THE USER TO PRINT THE LINE. DBUG1 ONLY SUPPLIES THE MESSAGE, IF REQUIRED, OR BLANKS.

3.2.4 DBUG1 DeBug SUBROUTINE (Cont.)

OPTIONS:

DBUG1 may be used with or without GEFRC IOEDIT. If IOEDIT code 2 is used to print the test message, it should be noted that the IOEDIT code 2 prints only 20 computer words (120 characters), while DBUG1 supplies 22 words (132 characters). Thus, if the print line specified by the IOEDIT list begin with the second word of the print line specified in the DBUG1 calling sequence, the message will be properly centered. In any case the message, if present, should always be the second line of the printed page.

3.2.5 COLUMN-BINARY/HOLLERITH CONVERSION

Purpose

To convert any number of 6-bit Hollerith characters to their 12-bit column-binary equivalents or vice-versa.

Usage

Calling Sequences

CALL HOLBIN (B,H,M) for Hollerith to binary.
CALL BINHOL (B,H,M) for binary to Hollerith.
Where: c (B)0-11 = first column-binary character
c (H)0-5 = first Hollerith character
c (M)0-35 = number of characters to be converted.
This routine uses 134 words
No error conditions.
Restrictions

None

Method

The subprogram converts each Hollerith character to column-binary by a direct reference to a 64-word table.

The subprogram converts each column-binary character to Hollerith by three direct references to the 64-word table. The first reference gives the class number and value of rows 12-3. The second reference gives the class number and value of rows 4-9. If the sum of the two class numbers is less than 4, then the sum of the value (a number from 0 to 63 inclusive) gives the location in the table where a third reference will specify the correct Hollerith character. If the sum of the two class numbers is not less than

3.2.5 COLUMN-BINARY/HOLLERITH CONVERSION

4, then the column-binary character is illegal and its Hollerith equivalent is octal 17, an ignore character.

The subroutine will fill out the last word of the result with spare characters if necessary.

3.2.6 NUMERIC FORMAT CONVERSIONS

Purpose

To perform binary-to-decimal and decimal-to-binary numeric format conversions in the same manner as the FORTRAN I/O conversion subroutine FRWD.

Usage

Calling Sequences

```
CALL  SFLBTD  (M,T,W,D,P)  for S.P. FL. PL. BIN. to FL. PT. DEC.
CALL  DFLBTD  (M,T,W,D,P)  for D.P. FL. PT. BIN. to FL. PT. DEC.
CALL  SFXBTD  (M,T,W,D,P)  for S.P. FL. PT. BIN. to FX. PT. DEC.
CALL  DFXBT   (M,T,W,D,P)  for D.P. FL. PT. BIN. to FX. PT. DEC.
CALL  SFLDTB  (M,T,W,D,P)  for S.P. FL. PT. DEC. to FL. PT. BIN.
CALL  DFLDTB  (M,T,W,D,P)  for D.P. FL. PT. DEC. to FL. PT. BIN.
CALL  SFXDTB  (M,T,W,D,P)  for S.P. FX. PT. DEC. to FL. PT. BIN.
CALL  DFXDTB  (M,T,W,D,P)  for D.P. FX. PT. DEC. to FL. PT. BIN.
CALL  INTBTD  (M,T,W)      for INTEGER BIN.      to DEC.
CALL  INTDTB  (M,T,W)      for INTEGER DEC.      to BIN.
```

Where:
- M = Location of binary number
- T = Location of decimal number
- W = Location of width of source number field
- D = Location of number of places to right of decimal point in source number
- P = Location of scale factor specifying number of significant digits desired to left of decimal point.

This routine uses 552 words.
No error conditions. On subroutine exit, the A-register contains the number of input errors encountered. This error count is also the contents of SYMDEF symbol BCDERS; it may, therefore, be referred in an IF statement.

3.2.6 NUMERIC FORMAT CONVERSIONS (Cont.)

Restrictions

The subprogram FIDO must be in memory. It consists of the double precision powers of 10 from 10^{-38} to 10^{+38}.

Method

The calling sequence combines the functions of a list and a FORMAT statement in specifying the conversion parameters. Each conversion routine is customized version of a routine within FORTRAN conversion routine FRWD.

All decimal fields begin with the left most character of the first word. If necessary, each binary-to-decimal conversion routine fills out the last word with ignore character (17 8).

3.2.7 RETRIEVE DATE

Purpose

To retrieve current date in one of six possible formats.

Usage

Calling Sequence:

CALL DATE 1 (N_{18}) for format September 10, 1968.
CALL DATE 2 (N_{7}) for format 10SEP68.
CALL DATE 3 (N_{8}) for format 09-10-68.
CALL DATE 4 (N_{8}) for format 09/10/68.
CALL DATE 5 (N_{6}) for format 091068.
CALL DATE 6 (N_{5}) for format 68253.

Where: N = Address of first word of field into which date is to be placed, left justified.
And Subscript = Number of characters that must be allowed for date.
This Routine uses less than 1,000 words.
No error conditions.

Restrictions

If this routine is called in a COBOL program, the receiving field N must be defined as a 01 or 77 level alphanumeric field.

3.2.7 RETRIEVE DATE (Cont.)

Method

3.2.8 .GBCD

Function

To convert a binary integer to its BCD equivalent.

Calling Sequence

CALL .GBCD

Description

This subprogram must be entered with the binary number in the A-register, right justified. The BCD equivalent is returned, right justified, in the combined AQ registers.

Core Required

26 words.

NOTE: The BCD number returned by this subroutine must not exceed ten digits. This subprogram isscontained in the System Subprogram Library, and is supplied to the user automatically at execution time.

3.2.9 .GBNRY

Function

To convert a BCD integer to its binary equivalent.

Calling Sequence:

CALL .GBNRY

Description

This subprogram must be entered with the BCD number right justified in the combined AQ registers. The binary equivalent is returned right justified in the Q register.

3.2.9 .GBNRY (Cont.)

Core required:

18 words.

NOTE: The BCD number to be converted must not exceed ten digits. This subprogram is contained in the System Subprogram Library, and is supplied to the user automatically at execution time.

3.3 CONTROL CARDS (Cont.)

Control cards are the "language" by which the system is instructed to perform various functions in a specific order. Some examples of these functions are: Compile, Execute, Assign Core Limits, and Assign and Release I/O Devices.

```
$       IDENT , NAME1,XX9999,XXX,99 TO 9XX0090000
0000000000 00000000  0000    000  00    0    0000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

16-20	Name of the program
21	, (comma)
22-27	Control Number
28	, (comma)
29-31	Your initials (always use three columns)
32	, (comma)
33-38	Job Identification
	33-34 Project Code
	35 Blank
	36 Run Code
	37 ReRun Code
	38 Blank
	39-44 Work Order Number
	45-48 Program Number
49-80	Blank

The Control Number and Job Identification fields correspond to similar fields on the "Computer Operations Work Request" form. The information for these fields will be furnished by the user's Lead Analyst or Supervisor.

The Job Identification field is broken into four basic subfields as shown.

The Run and Re-run codes (columns 36 - 37) are defined on the back of the Computer Operations Work Request. The original submission of any job should have a re-run code of zero (0) in column 37 of the $ IDENT card. On subsequent submissions of the same job, column 37 should contain the appropriate re-run code.

3.3 CONTROL CARDS (Cont.)

```
$       FORTRAN OPTIONS
0000000000 000000 000 0000000000000000000000000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

This is a required card which indicates to the computer the user's desires for compile activities and options. There are many options as described in CPB-1688 CONTROL CARD Manual. (Page 76)

```
$       COMMENT ANY COMMENTS
0000000000000 000 0000000 0000000000000000000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

This card is used by the programmer to communicate to the operator through the on-line typewriter. When pertaining to an execution following other activities, this card should be placed directly behind the $ EXECUTE control card.

```
$       INCODE  IBMP
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

This card is required when using special characters in the IBM Character Set. When no special characters are used this card is not necessary; however, it is recommended that it be used whenever using the IBM Character Set. This card must not be used with the GE Character Set.

3.3 CONTROL CARDS (Cont.)

```
$ ENDJOB
```

This is a required card that specifies that this is the end of cards for a particular job. It must be the next to last card in any job stream and is followed by the **EØF card.

```
**EOF
```

This card is required to specify the end of the job stream to the Card Reader as did the $ ENDJOB card for the system.

```
$ EXECUTE DUMP
```

The $ EXECUTE DUMP control card is a very powerful card which can perform many services for the user. For example, it can load OBJECT programs from the System Library and turn on sense switches. As the user's programming matures and becomes more complex he will need to know these options. Consult GE-635 Manual, CPB-1518 GECOS II Reference Manual, Section 5, Basic Input Deck, for details. The beginning programmer will only want to use the DUMP option. This will cause a dump of the program to be taken if an error occurrs during execution.

3.3 CONTROL CARDS (Cont.)

```
$ LIMITS TIME,MEMORY,BLOCK COMMON,PRINT LINES

00000000000 00 000 00000 000000000000 0000 00000 00000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

This is a required card and must be placed directly behind the $ EXECUTE control card. It has four entries designating time, memory, block common and print lines. Time is written as an integer designating hundredths of an hour. This processor time and not the elapsed time the user program will be in 635 memory. This time is the maximum time that the user will allow his job to run, assuming that if its takes longer the program has an error, such as a loop. This field is three digits with a maximum time of 9.99 hours. Enter as small a number as possible in order to avoid excessive processor usage.

The second field is memory or core size. Once again, use a small a number as feasible in order to conserve core for other programs.

The third field is block common. The field indicates to the loader how much common area can be shared. A discussion of this can be found in CPB-1688 with the $ LIMITS control card.

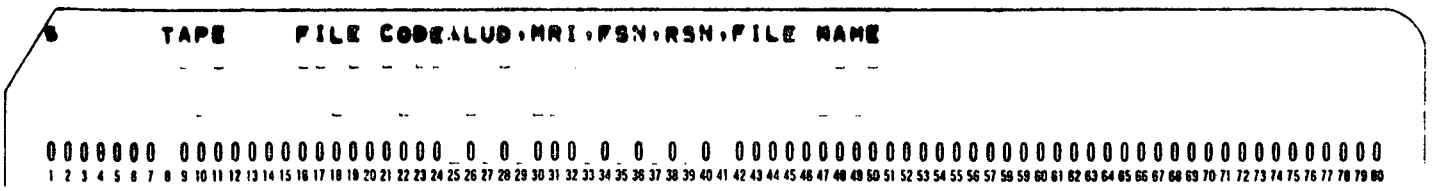
The fourth and last field is the maximum number of print lines. This is not as critical as time or memory as to exactness of size, but keep in mind that this field is for the purpose of preventing the printer from being indefinitely tied up in case of a print loop, and make a reasonable extimate of print lines accordingly. However, be sure to allow enough for a dump if necessary. The maximum value for executes is 10,000 lines.

A sample of a limits card might be:

```
$ LIMITS 09,9000,15000

00000000000 00 0 0 0 00000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

3.3 CONTROL CARDS (Cont.)

This sample would allow 3 minutes (.05 hours) processor time, 9K of core, no block common, and up to 5000 lines of print. In making out the Computer Operations Work Request form, the extimates required in the Remarks Section would be 9K, the number of tapes required, and an estimate of the elapsed time.



$ TAPE card. This card is used to tell the system how to use the tapes needed and what disposition to make of the tape after completion of the job.

FILE CODE - This is the code specified in the user program by a READ or WRITE statement. Example: READ (11,000) list
11 is the file code

WRITE (12,000) list
12 is the file code.

Although the system allows the file code designation to be alphanumeric or alphabetic, FORTRAN does not allow the file code to be anything but an unsigned integer constant from 1 through 40.

LOGICAL UNIT DESIGNATOR - (LUD) This is a code used to indicate to the operating system the disposition of any associated tapes. It is normally a three digit code, but may have up to five digits if needed. It is separated into three parts.

SYMBOLIC CHANNEL DESIGNATOR - Any single alphanumeric character. This will have no function if the tape is used in a single activity. If the tape is to be used by multiple activities this designator and the numeric designator must be the same on each tape card for this file throughout the job stream to prevent the system from reassigning the tape to a different tape unit for each activity.

NUMERIC UNIT DESIGNATOR - This is a one or two digit decimal numeric designation which in combination with the Symbolic Channel Designator applies uniquely to the tape file assigned by the $ TAPE control card.

3.3 CONTROL CARDS (Cont.)

NOTE: If the programmer wishes to use the same file in different activities, he must assign the same channel designator and unit designator to that file on the tape cards in the different activities. The file code does not have to be the same.

DISPOSITION CODE - This third part of the Logical Unit Designator is a one or two digit code. The first digit indicates use of the tape file in a normal termination. The second digit indicates use of the tape file in the event of an abnormal termination. The codes are:

S - Save for further activities in the same job stream.

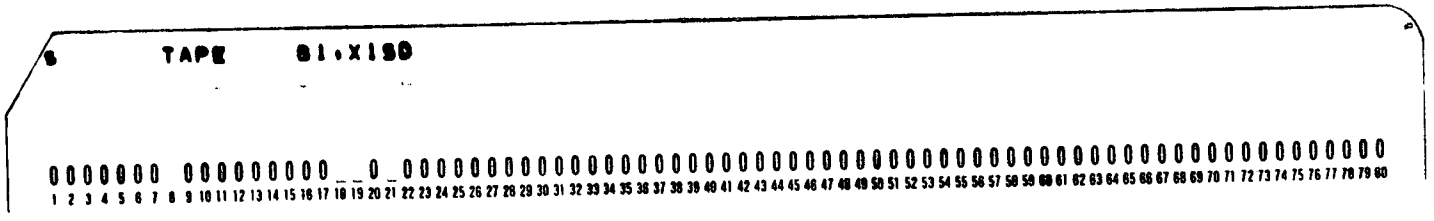
C - Continue. The computer operator must dismount the tape, remove the write ring and re-mount the tape before it can be used in the next activity. This disposition code should not be used unless absolutely necessary to preserve the integrity of a file.

D - Dismount. The tape will dismound and the operator will label it with a label supplied by the programmer.

R - Release. Releases the tape back to the system for scratch use. This is the disposition used whenever the job is finished with a given tape.

The disposition code may be two digits in length as previously stated. For example: SR indicates to save for the next activity in a normal termination or release in case of an abort.

Example:



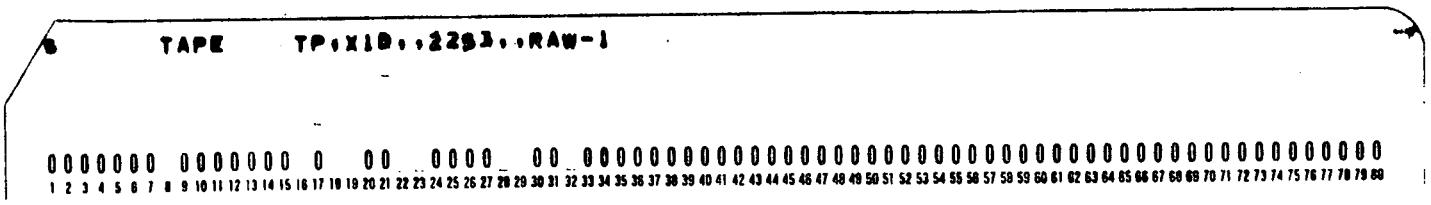
3.3 CONTROL CARDS (Cont.)

B1 represents the File Code.

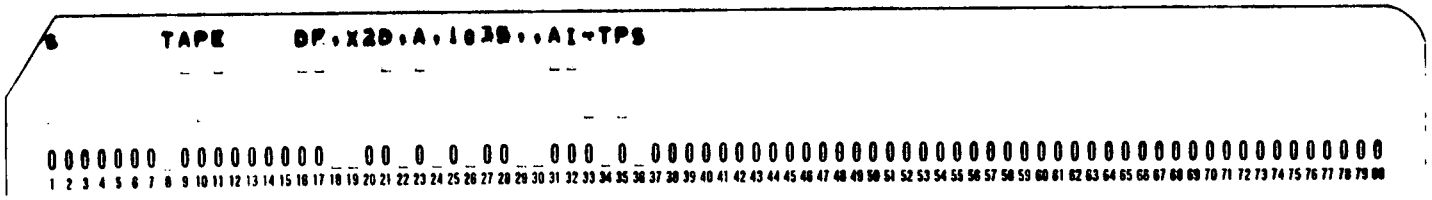
X1SD represents the Logical Unit Designator.

- X - Symbolic Channel Designator
- 1 - Numeric Unit Designator
- S - Save for next activity
- D - Dismount (Will occur only on abnormal termination)

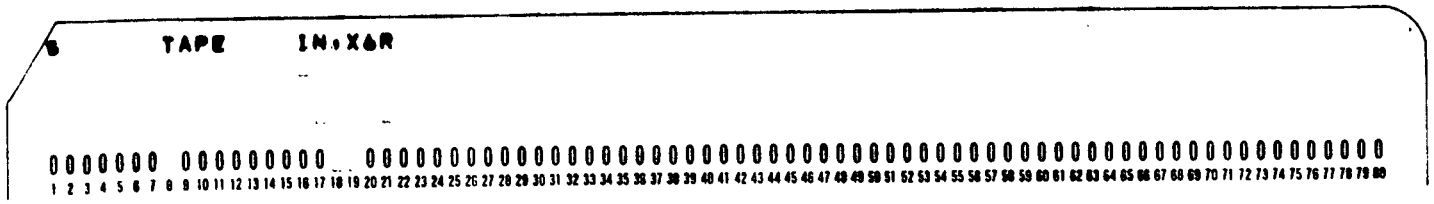
Sample Input Tape Card



Sample Output Tape Card



Sample Scratch Tape Card



MULTI-REEL INDICATOR (MRI) COL-23-25 Commonly referred to as Unit Switching. Any non-blank character in this field will cause a second tape unit to be assigned to the file. This field is used for files containing two or more reels of tape. The tapes are mounted alternately on the two tape units. When a multi-reel file is not used, this variable is omitted; however, the comma denoting end-of-file must be included if other fields follow.

3.3 CONTROL CARDS (Cont.)

Example:

```
$       TAPE     A1,X1R,X2R,9999,,RAW-INPUT-TAPES

0000000 000000000  00  00 0000  00 0000  0 000 0000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

FILE SERIAL NUMBER (FSN) - This is the tape library number of a tape file and the primary input file label checking parameter. If this value cannot be specified, use 99999, is which case the field will be ignored during label checking. This field is not required and may be omitted for output files and input files supplied by the user.

REEL SEQUENCE NUMBER (RSN) - This is the number of the reel at which processing is to begin. This is a label checking parameter, but is omitted unless processing does not begin on the first reel of a labeled, multi-reel input file. If other fields follow, a comma denoting end-of-field must be present.

FILE NAME - The name assigned to the file for external indentification. It is used to issue mounting instructions to the operator. This field can be more than 12 characters long. Imbedded blanks may not be used.

The $ OPTION card is a loader control card used in execution setups is not needed during compilation. It should be placed on the front of the object deck.

Example:

```
$       OPTION  OPTIONS

0000000000 0000000 000 0000000000000000000000000000000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

NOTE: There is no need for OPTION GMAP and OPTION COBOL cards as these functions are now handled within GECOS.

3.3 CONTROL CARDS (Cont.)

Input/Output Tapes and the Computer Work Request

All tapes used on the GE-635 should have external labels with the exception of scratch tapes. These labels are to identify the tape to the Operations personnel. The wording or numbers on these labels should appear on the Computer Work Request and $ TAPE cards.

For example:

An execution to be run on the GE-635 using a Library Tape #1550 labeled "SORTED-INPUT", another input tape from the user labeled "DUMMY DATA", one scratch tape and one output tape labeled "FINAL OUT" should be submitted as follows:

$ TAPE CARDS

```
$       TAPE      A1,X2D,,1150,,SORTED-INPUT
$       TAPE      A2,X3D,,XXXX,,DUMMY-DATA
$       TAPE      A3,X4R,,XXXX,,SCRATCH
$       TAPE      A4,X5D,,XXXX,,FINAL-OUT
```

The Computer Work Request should complement these tape cards and tape labels as illustrated in Figure II.3.

```
$       FFILE   FILE CODE,PARAMETERS
```

The $ FFILE control card is used to alter a file control block (FCB). One FCB is altered by each $ FFILE card.

Example:

```
$       FFILE   02,NOSRLS,NSTDLB,FIXLNG/200,ERRXIT/ERRXI,LODENS,MLTFIL
```

COMPUTER OPERATIONS WORK REQUEST

CLASSIFICATION	PRIORITY
U C S	700

TIME IN	TIME OUT	REQUESTOR'S NAME	PHONE
		G. POE	7-8471

RECEIVED FROM	RECEIVED BY	JOB	√
		KEYPUNCH AND VERIFY	
		EAM	
		1004	
		7702	
		635	X
		4020	

JOB NO.	PROGRAM	CONTROL NO.
	EXEC1	S P 1 0 0 0

ITEMS SUBMITTED TO BE RETURNED

DESCRIPTION	SECURITY CLASS	COPIES SUBMT.
TAPE "DUMMY DATA"	U	1
LABEL "FINAL OUT"	U	1
PROGRAM DECK & CONTROL CARDS	U	1

INPUT TAPES

REEL	CONTROL	UNIT
Reel #1550 SORTED INPUT		
DUMMY DATA		

OUTPUT TAPES

REEL	CONTROL	UNIT
SCRATCH TAPE		
FINAL OUT		

ITEMS TO BE GENERATED

DESCRIPTION	SECURITY CLASS	COPIES REQUEST	COPIES GENERATE

RERUN CODES

1	2	3	4	5	6	7	8	9	A	B	C	D	E

JOB IDENTIFICATION

10 Digit Work Order Number - Project Code

REMARKS:

20 K

06 Min.

4 TAPES

FIG. 3.1

3.3 CONTROL CARDS (Cont.)

The several parameters available are described in the GE-635 Control Cards Manual, CPB-1688. This card is used to bypass or alter data in the FCB so that a non-standard file may be used by the system.

```
$ FILE    FILE CODE,LUD,ACCESS CODE

$ MASS    FILE CODE,LUD,ACCESS CODE
            U     U   U    U     U
```

The $ FILE control card is used to set up a file on the fastest mass storage device available. The $ MASS control card is used to set up a file on the slowest mass storage device available. The Logical Unit Designator is changed only in the number of allowable disposition codes.

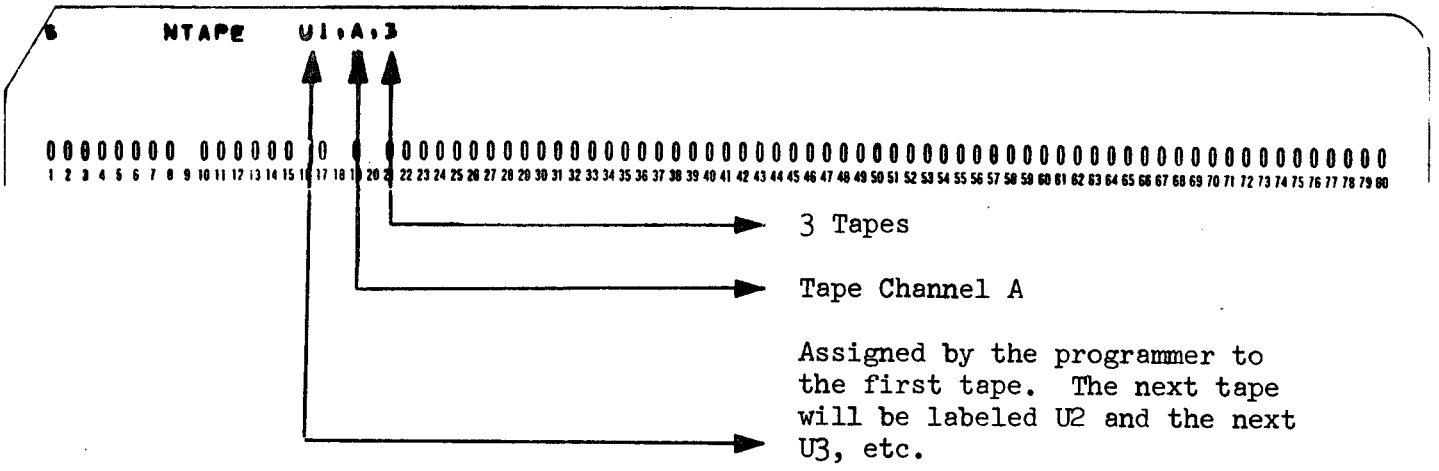
R - Release to system.
S - Save for subsequent activity.
C,D - Have no meaning on the $ FILE and $ MASS control cards.

The Access Code contains a numeric and an alphabetic designation. The numeric designation specifies the number of links (3840 words equals 1 link) to be allocated to the file. The alphabetic portion is either an R or an L where R specifies a random file and "L" specifies a linked file. Normally, the programmer will use a linked file which is address sequentially, as a tape is read or written. Records in a random file may be addressed in any order, but reading or writing these files require special programming.

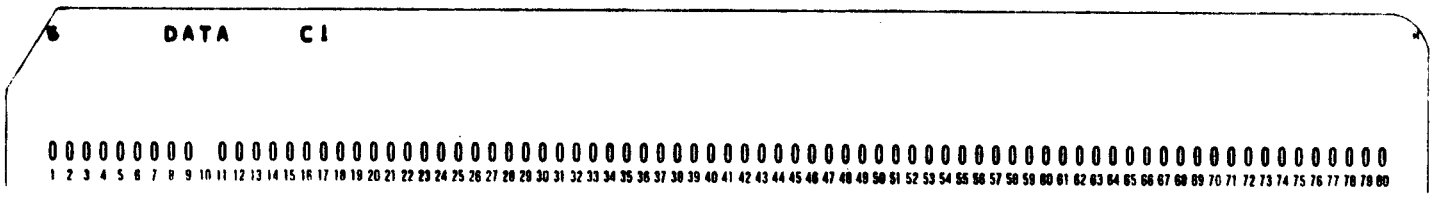
```
$ SELECT  KSCPERM/PROG1

$ USERID  KSC,KSC
```

The $ USERID and $ SELECT control cards are used to obtain programs from the Perm File for execution. The syntax of each is as shown below with the exception of "PROG1" which must be the name of the program desired.

3.3 CONTROL CARDS (Cont.)



This control card is used by GEIN to request a number of utility tapes to be assigned to the tape channel in the Operand field.



This card is used to write files onto temporary linked system storage for input. These files may **not** be written on by the user. Although these files are removed from the system after termination of the job, it is possible to save them from one activity to another by the use of a file code which has a numeric or asterisk for the second character and adding an "S" to the file code field. (See CPB-1688, page 23 for available options.)

3.3 CONTROL CARDS (Cont.)

The control cards previously mentioned are most commonly used by most programmers. There are many more, offering the programmer a wide variety of options and control. These are explained in the Control Card Manual, CPB-1688.

SECTION 4 SYSTEMS INFORMATION BULLETIN

Systems Information Bulletin No's. 10, 12, 16 and 17 have been deleted as they pertain to GECOS II only.

The reader should be aware of bulletins issued subsequent to the revision date of this manual.

Subsequent bulletins will be incorporated in the next general revision of this manual.

4.1 SYSTEMS INFORMATION BULLETIN #1

SUBJECT: Drum Capability FROM: FEC/NASA Systems Programming

Drum Capacity: The drum has a storage capacity of 786,432 words. These words are grouped into 384 bands, with each band containing 32 units known as data blocks. Each data block contains 64 words. Each drum has 204 links of 3840 words per link. There are two drums available for use by the programmer. The programmer has all links on one drum and 84 links on the other drum.
Drum Access Time: The drum has an average access time of 17 milliseconds and a maximum access of 34 milliseconds.
Drum Usage: The programmer has available 2 methods of storing information on the drum. He can store it as a random or as a linked file. Random files are handled thru CALL READ or CALL WRITE options of GEFRC. These files are referenced by a block number relative to the beginning of the file. Each block is 64 words long. Example for putting 200 words of data on a drum:

```
        CALL WRITE (FCB, DCW, CC)
          .
          .
          .
DCW DEC 1
    IOTD BUFFER,200
```

This data would be placed in 4 consecutive blocks starting at block 1.
For linked files, standard systems format must be used and FORTRAN or GEFRC standard routine can be used.
The storage reserved on the drum is done by a $ DRUM card. The reservation is made by number of links needed. Each link represents 3840 words. For a random file, the Letter R follows the number of links. For a linked file, the letter L follows the number of links.

4.2 SYSTEMS INFORMATION BULLETIN #2 7/25/67

SUBJECT: END-OF-FILE PROCEDURES FROM: NASA/FEC Systems Prog.

The following procedure is recommended for inclusion in all pro-grams to prevent false end-of-file recognition:

When a record is tested by a program and the EOF condition is indicated, the program should try to determine whether this is a valid EOF condition. This can be accomplished by testing the substatus field of the status word for octal 17. If octal 17 is not present, the program should assume a false end-of-file has been indicated and continue processing.

It is further recommended that all programmers use only octal 17 to indicate EOF.

4.3 SYSTEMS INFORMATION BULLETIN #3 7/31/67

SUBJECT: DISC USAGE FROM: NASA/FEC Systems Programming

When disc is used, the access mode of a file may be either linked (L) or random (R). Linked files are sequential in nature and may be thought of and used interchangeable with tape files if GEFRC is used. The element of access when reading or writing linked files is a 320-word record. Access to linked files are handled by the GEFRC except when passing files between activities; then the user must rewind the file if it is to be positioned for the next activity.

Access to random files are the responsibility of the user. As part of his calling program, the user must specity the relative block number within a referenced file. The relative block size for disc is 40 words/blocks.

Random files are handled thru CALL READ or CALL WRITE options of GEFRC. Example for putting 200 words of data on a random disc file:

```
        CALL WRITE (FCB, DCW, CC)
          .
          .
DCW     DEC 1
        IOTD BUFFER,200
```

This data would be placed in 5 consecutive blocks starting at block 1.

4.3 SYSTEMS INFORMATION BULLETIN #3 (Cont.)

The READ or WRITE routines should be followed by CALL WAIT. When the device is random file for disc and the last consecutive block is assessed, an end-of-file condition will be indicated and the end-of-file branch in CALL WAIT will be executed. Transmission beyond this last consecutive block will not be performed until the block count is increased. The last consecutive block is a relative block address whose last 5 bits are 11111. Therefore the programmer must know that for each 32 blocks the systems treats this arrangement as a physical file on disc.

For allocation, the storage reserved on the disc is done by a $ DISC card. The reservation is made by number of links needed. Each link represents 3840 words. For a random file, the letter R follows the number of links. For a linked file, the letter L follows the number of links.

4.4 SYSTEMS INFORMATION BULLETIN #4 8/2/67

SUBJECT: Compressed Symbolic Prog. Tapes FROM: NASA/FEC Systems Prog.

If a compressed symbolic program deck is very large, it may be desireable to put it on tape and assemble the program from tape rather than cards. To do this, two system files are redefined as tape files rather than disc files. These files are:

G*	GMAP Source file
K*	Compressed deck file

For very large programs it may necessary to redefine the *1 scratch file used for assemblies as a tape file to avoid aborts caused by exceeding the amount of disc storage allotted to this file. It may also be necessary to expand the standard limits of 24000 memory locations assigned to the GMAP assembler. This may be done by placing a LIMITS card behind the GMAP card.

The following deck set-up is used to go from a symbolic program compressed deck to a symbolic program compressed tape.

```
$   IDENT
$   GMAP      COMDK
$   LIMITS    20,32000,0,9999
$   TAPE      K*,X1D,,XXX,,OUTPUT-CMDK-TAPE
$   TAPE      *1,X2D,,XXX,,SCRATCH
    .
    .         COMPRESSED PROGRAM DECK
    .
```

4.4 SYSTEMS INFORMATION BULLETIN #4 (Cont.)

```
$       UPDATE          LIST
$       INCODE          IBMF
$       ALTER           1,1
        .
        .               ALTERS TO PROGRAM
        .
$       EXECUTE
etc.
```

NOTE: The output Compressed Program tape will not be generated unless there is at least one ALTER card. Also, only one INCODE card needed for the ALTER deck.

The next deck set-up is used to assemble from tape without generating a new compressed program tape.

```
$       IDENT
$       GMAP
$       LIMITS          20,32000,0,9999
$       TAPE            G*,C1D,,XXX,,COMDK-TAPE
$       TAPE            *1,X2D,,XXX,,SCRATCH
$       UPDATE          LIST
$       INCODE          IBMF
$       ALTER           1,1
        .
        .
$       EXECUTE DUMP
etc.
```

A new Compressed Program tape may be created from the old one with the following card set-up.

```
$       IDENT
$       GMAP            COMDK
$       LIMITS          20,32000,0,9999
$       TAPE            G*,X1D,,XXX,,OLD-COMDK-TAPE
$       TAPE            K*,X2D,,XXX,,NEW-COMDK-TAPE
$       TAPE            *1,X3D,,XX,,SCRATCH
$       UPDATE          LIST
$       INCODE          IBMF
$       ALTER           1,1
        .
        .
$       EXECUTE         DUMP
etc.
```

4.5 SYSTEMS INFORMATION BULLETINE #5 8/29/71

SUBJECT: New Timer Interrupt Rates for RTICOC#2 FROM: Systems Prog.

A modification has been installed in the Timer Adapter on RTIOC#2 to include six new timer interrupt rates. For these new rates the high-low range toggle switch must be in the low position or up. For the old rates the switch must remain in the high position or down.

An additional bit has been added to the status word to indicate the position of the high-low range toggle switch. Note that incorrect setting of the switch will result in obtaining incorrect status bits.

The following tables show the new and old interrupt rates and indicate which status bits are on in each case:

1. New rates (Switch in low or up position)

RATES	STATUS BITS ON
0.25 per sec.	25,27
0.5	25,29
1	25,31
2	25,32
4	25,33
8	25,34

2. Old rates (Switch in high or down position)

RATES	STATUS BITS ON
12 per sec.	26
16	27
24	28
32	29
48	30
64	31
124	32
256	33
512	34
1024	35

The new rates should be included in RTIOC#1 no later than 11 September 1967.

4.6 SYSTEMS INFORMATION BULLETIN #6 9/21/71

SUBJECT: Defining Paper Type for Printer With $ REPORT FROM: Systems Programs

The $ REPORT Control Card can be used to Define a Specific Printer Form When Using SYSOUT. The typewriter message now includes the

4.6 SYSTEMS INFORMATION BULLETIN #6 (Cont.)

Job and activity numbers.

Card Format:

```
1    8          16
$    REPORT     (,PR,XX,N
```

Where XX is a 12 character or less paper form identification (No blank characters) and N is the number of plys of paper. The left parenthesis "(" is used for the IBM character set now in use at KSC (Octal Code 74).

EXAMPLE:

```
$  SNUMB 20179
.
.
.
$  EXECUTE
$  REPORT (,PR,STANDARD,3
.
.
```

Typewriter messages to operator:

```
*  Change Form Print 10 Standard 3 Ply    20179-01
                     (7)
```

At the conclusion of the print activity:

```
*  Change Form Print 10   1 Ply    20179-01
                     (7)
```

At which time the form can be changed back to the standard one. Use of this control card has been vertified for the execution report only.

4.7 SYSTEMS INFORMATION BULLETIN #7 9/29/67

SUBJECT: Systems ID on a Master Mode Dump FROM: Systems Prog.

The SYSTEM ID has been changed to reflect the System, Version on the SDL, and any modification to this Version. The following examples will describe the changes with the implementation of SDL-11.

EXAMPLES: (With SDL-11)

1. SYSTEM ID A1V1M1
2. SYSTEM ID A2V2M2
   a. Characters 1 and 2, A1 or A2, will be typed in at edit or boot time by the operator. A1 and A2 will represent Systems No. 1 and No. 2.
   b. Characters 3 and 4, V1, V2; Vn, will represent the Nth Version of the SDL.
   c. Characters 5 and 6, M1, M2; MN, will represent the Nth Modification of (b).

4.7 SYSTEMS INFORMATION BULLETIN #7 (Cont.)

EXAMPLES: (With SDL-10)
1. SYSTEM ID A1
2. SYSTEM ID A2
   The SDL number can be obtained from Cell 5738.

4.8 SYSTEMS INFORMATION BULLETIN #8 9/29/67

SUBJECT: Banner Page Modification FROM: System Prog.

The GECOS SYSTEM IDENT on the banner page of all SYSOUT Listings has been changed to reflect the System, Version of the SDL, and any modification to this Version. The following examples will describe the changes with the implementation of SDL-11.

EXAMPLES: (SDL-11)
1. GECOS SYSTEM IDENT A1V1M1,
2. GECOS SYSTEM IDENT A2V2M3,
   a. Characters 1 and 2, A1 or A2, will be typed in at edit by the operator. A1 and A2, will represent System No. 1 and No. 2.
   b. Characters 3 and 4, V1, V2; Vn, will represent the Nth Version of the SDL.
   c. Characters 5 and 6, M1, M2; MN will represent the Nth modification of (b).

EXAMPLES: (SDL-10)
1. GECOS SYSTEM IDENT A2 ,
2. GECOS SYSTEM IDENT A2 ,

4.9 SYSTEMS INFORMATION BULLETIN #9 12/5/67

SUBJECT: RERUNS OF ABORTED PROGRAMS FROM: NASA/FEC Systems Prog.

Slave programs which have the following aborts should be rerun by the operator before being returned to the programmer.

(1) B9 An entry made to an overlay is erroneous. Proable hardware failures.
(2) F5 Connect fault. (Hardware failure).
(3) F6 Memory parity fault. (Hardware fuilure).
(4) Q4 Machine error or unexpected error to Fortran.
(5) J9 Input read error.
(6) JG System card reader inseparable.
(7) JI Simulator (9SA0) not configured.

4.9 SYSTEMS INFORMATION BULLETIN #9 (Cont.)

The following slave aborts may be caused by programmer error, software failure, or tape error. Programs having these aborts should be retired only once.

(1) A1 Parapheral Assignment Table buffer size exceeded.
(2) L1 Missing subroutine required.
(3) L3 Fatal error encountered during loading.
(4) U3 Hardware error detected during UTILITY activity.

NOTE: These are some GEFRC errors which should also be re-run However, the error messages do not come out on the typewriter at this time. This will be remedied under SDL-12.

4.10 SYSTEMS INFORMATION BULLETIN #11 1/11/68

SUBJECT: One Card Memory Dump on Tape FROM: IN-DAT-41/867-7334

To dump memory with the one-card dump:

0. Turn off RTIOC channel (0) on lower memory controller.
1. Mount a scratch tape on 2-0 and ready it.
2. Push "INIT" button on Console.
3. Put one-card dump in system card reader.
4. Put card reader in operate.
5. Push "BOOT" button on console.
6. Dump will immediately come out on tape 2-0. When tape stops, dump in complete. Tape will not rewind automatically.
7. Label,tape (malfunction type, date, time of day). Turn tape, pertinent typewriter console sheets, and malfunction report over to Systems Programs Section, IN-DAT-41.

4.11 SYSTEMS INFORMATION BULLETIN #13 1/18/68

SUBJECT: Tape Write Capability FROM: IN-DAT-41/867-7334

A study was recently made to determine the maximum length record that could be written on the GE-635 at various write rates. Primarily because of the interest generated when typing to write large records with the Mission Program, the rates chosen for this study were based on RTIOC interrupt rates.

The test program used for this study has essentially NO overhead. Thus the statistics contained herein should be viewed as optimistic. Attachment A shows the results of the analysis.

4.11 SYSTEMS INFORMATION BULLETIN #13 (Cont.)

A rough plot of the data contained in the attachment indicates that the average start and stop time of the 770 tape handler, running in a non-continuous mode, is about 7-8 milliseconds.

ATTACHMENT A

The maximum length record that can be written during:

1/128th	OF	A	SECOND	IS	0	WORDS
1/64th	"	"	"	"	147-153	"
1/48th	"	"	"	"	252-257	"
1/32nd	"	"	"	"	460-465	"
1/24th	"	"	"	"	668-673	"
1/16th	"	"	"	"	1085-1089	"
1/12th	"	"	"	"	1500-1508	"

4.12 SYSTEMS INFORMATION BULLETIN #14 2/20/68

SUBJECT: Standard Labels - COBOL FROM: Systems Prog. Pritchard/8470

All output files generated in COBOL use Standard system format which calls for standard labels. Under SDL-11 requirements, the COBOL file description statements must now contain the following information for label processing:

1. All file description statements for standard systems format should use the "LABEL RECORDS ARE STANDARD" clause.
2. All file description statements for output files should also use the "VALUE IDENTIFICATION IS" clause.

Programs now in production which do not follow these rules may have to be recompiled if they result in G1 aborts.

4.13 SYSTEMS INFORMATION BULLETIN #15 2/18/68

SUBJECT: COBOL Tape Cards FROM: Systems Prog. Pritchard/8470

To avoid improper mounting of tapes and subsequent G1 aborts there must be adequate communication between the program and the operator via tape cards. In general, the following requirements are needed:

1. The reel serial number should contain either the actual reel number or the number 9999.
2. The last field, which should correlate with the external tape label, should contain enough information so the operator will always mount the correct tape.

4.13 SYSTEMS INFORMATION BULLETIN #15 (Cont.)

When possible, for program testing and in documentation the following procedures should be used to identify tapes:

A. MASTER FILES have a file and cycle designation.
Example - FILE 56 CYCLE 1.

The combination of file and cycle number is unique. Therefore, the following format is recommended for tape cards referencing MASTER FILES:

```
CC1         8          16
$           TAPE       A1,B2D,,9999,,FILE56CY1
```

The fourth parameter should ALWAYS be 9999 as the reel serial number will change from week to week. The sixth parameter is unique and on the outside label of the tape.

B. WORK TAPES should be identified by a unique file number.
Example - 7296.40 where 7296 refers to the program number it is created in and 40 is the unique file number. The file numbers should be designated as follows:

20-29	file created by card to tape
40-49	file to be forwarded to succeeding run within system
50-59	file to be saved in library
60-69	file to be forwarded for print/punch

Example: The following tape card formats could apply for two print tapes from job #7226.

```
CC1         8          16
$           TAPE       A1,B1D,,9999,,7226.60
$           Tape       A2,B2D,,9999,,7226.61
```

The reel serial number is never known from one week to the next. The program and file numbers will never change. Therefore, if programmers want to save tapes from test runs their external tape labels should include the program and file numbers. Programmers should also refer to these labels when making up the cards.

C. SCRATCH TAPES should have the following format:

```
CC1         8          16
$           TAPE       A1,B2R
```

4.14 SYSTEMS INFORMATION BULLETIN #18 1/22/69

SUBJECT: F-7 Abort Code Definition Change FROM: Systems Prog. FEC/460-JDR

There has been a recent GE-635 Computer hardware modification that is reflected in the F-7 Abort indication.

PREVIOUSLY - If an illegal instruction operation was executed, the results could not have been predicted. In some cases this would have had no effect on a program other than the execution of an apparent "NOP" instruction.

CURRENTLY - The execution of any illegal instruction operation will cause an F-7 Abort.

DEFINITION CHANGE - The new difinition for an F-7 Abort Code is as follows:
F-7 = Illegal OP-Code Fault (includes Zero-OP)

SPECIAL NOTE - Some previously checked-out production programs may now abort with an F-7 indication because of the recent hardware modification.

4.15 SYSTEMS INFORMATION BULLETIN #19 2/4/69

SUBJECT: GECOS-III Real-Time Checkpoint and Rollback Facility (Version 0).

GENERAL:

A. Description

The GECOS-III software system has been modified to provide Real-Time programs with the ability to quickly recover in the event that some error causes the Real-Time system to abort.

Chekpoint Logic - The Checkpoint software is designed to save the total 635 environment existing at the time of the checkpoint operation (call this time "T_c") and return control to the calling program.

Rollback Logic - The Rollback software is designed to:

(1) Load itself into memory from tape
(2) Completely re-establish "T_c" 635 environment
(3) Return control to the Real-Time program as though a checkpoint operation had just completed.

4.15 SYSTEMS INFORMATION BULLETIN #19 (Cont.)

NOTE: Total machine environment re-established after Rollback except for positioning of capture tapes, and/or batch job allocated tapes.

B. Recovery Timing

The recovery time estimates for the Rollback operation are as follows:

Rollback (Reload memory)	10 seconds
RT Program Tape Positioning	20-30 seconds
RT Program Final Initialization	5 seconds
Total Recovery Time	35-45 seconds

PROGRAMMING INFORMATION:

A. Usage

Real-Time Checkpoint is implemented by the Real-Time program via the execution of a new Real-Time master mode entry, "MME RTCHEK" (OCTAL 400027). The MME RTCHEK must be issued following the execution of the MME RTMODE and prior to RT-IOC I/O operation. (NOTE, if MME RTMREL is used, locate the MME RTCHEK following the MME RTMREL.)

All tape positioning/labeling should be performed following the Checkpoint call (because Rollback will return control to the instruction following the MME RTCHEK).

All program initialization that can be performed prior to the MME RTCHEK should be thus done to decrease recovery time after initiating a Rollback (ie: loading of coefficients should be done prior to the MME RTCHEK. Also, the coefficient tape, or any other non capture file, should be releases before the MME RTCHEK execution).

B. Restrictions

(1) The Real-Time program is responsible for positioning, termination and/or replacement of the capture tapes at Rollback time.

(2) The Q-Register will contain a return code in bit positions 34-35 as follows:

Bit 34	Bit 35	Return Code Definition
0	0	Checkpoint Return-Successful
0	1	Checkpoint Return-Failure
1	0	Rollback Return-Successful

4.15 SYSTEMS INFORMATION BULLETIN #19 (Cont.)

There will be no un-successful Rollback return code.

Bit positions 0-33 of the Q-Reguister will not be affected.

(3) The Checkpoint file is fixed as channel 2, unit 6 and need not be assigned/allocated by the Real-Time program.

OPERATIONS INFORMATION:

A. Checkpoint

The Checkpoint operation will require a tape (with write-ring in) to be mounted on channel 2, unit 6.

The Checkpoint tape will be automatically unloaded with an operator's message (on console) indicating success or failure of the Checkpoint operation. (A failure indication means only that the Checkpoint tape was bad, or that the handler malfunctioned, and that a Rollback operation is therefore not possible.)

When the Checkpoint operation is successful, a typewriter message will inform the operator as to switch settings for a Boot Load from the Checkpoint tape (see Rollback Boot procedure).

The Checkpoint tape should be held until the current Real-Time test is completed.

B. Rollback

(1) Rollback Initiation Conditions - The 635 operator will initiate Rollback under one of the following conditions:

a. The Real-Time program has been aborted, terminated or otherwise lost and restart is desired.

b. The Real-Time program needs to be re-started (but, Real-Time initialization remains the same).

c. A need to reload a previously checkpointed system (eg: GNEL test each morning).

4.15 SYSTEMS INFORMATION BULLETIN #19 (Cont.)

(2) Initiation Procedure (For Rollback Boot):

a. Set Boot Load Switches according to values typed by Checkpoint operation.

b. Ready Checkpoint Tape on channel 2, unit 6.

c. Depress Console Reset button.

d. Depress Console Initiate button.

e. Depress Console Boot Load button.

f. Correct Date/Time as soon as slave is running.

(3) Successful/Failure Indicators:

Rollback will attempt to load the Checkpointed system and will type a message as to success or failure. If failure occures, Normal Boot for the system will be required. If successful, the Checkpoint tape should be held for additional Rollback requirements.

(4) Restrictions:

a. Rollback will lose all batch work in the system.

b. No SYSOUT recovery will be attempted.

c. Rollback will not check peripherals assigned to the Real-Time program at Checkpoint time.

NOTE: Real-Time peripheral tape requirements will be identical to the requirements at the time the Checkpoint was taken and positioning will be the responsibility of the Real-Time program. (If the Real-Time program was in execution and tapes have not been unloaded, DON'T TOUCH, replace any unloaded tapes with ready scratch tapes BEFORE starting the Rollback Boot procedure.

4.16 SYSTEMS INFORMATION BULLETIN #20 2/12/69

SUBJECT: Usage of MME GEINOS Under GECOS-III
FROM: IN-DAT-41 Truitt/8767

Programmers using MME GEINOS must delay until I/O is complete before testing bits in the status return words. Under GECOS-II, the I/O may have been completed before the software returned to the slave program, but the software in GECOS-III has been speeded up so that return to the slave program is much faster. Thus, the status return words may still be zeroed out when the slave program tests them. Programmers may delay processing until I/O is complete by using a MME GEROAD or a MME GERELC statement. Contact IN-DAT-41 for further information.

4.17 SYSTEMS INFORMATION BULLETIN #21 2/17/69

SUBJECT: Named Device Capability Under GECOS-III
FROM: IN-DAT-41

This Systems Information Bulletin is superceded by Systems Information Bulletin #31, dated October 17, 1969.

4.18 SYSTEMS INFORMATION BULLETIN #22 4/8/69

SUBJECT: Perm-Save and Perm-Restore FROM: IN-DAT-41

A FILSYS activity must be run to accomplish either the saving or restoring of the permanent files on GECOS-III, SDL-1 (see FILE SYSTEM manual CPB 1513).

PERM-SAVE

```
$     IDENT
$     FILSYS
$     PRIVITY
$     TAPE    PS,X2D,,99999,,PERM-SAVE
SAVEMAST LISTOPT/YES/
$     ENDJOB
```

PERM-RESTORE

```
$     IDENT
$     FILSYS
$     PRIVITY
$     TAPE    PR,X3D,,99999,,PERM-RESTORE
RESTOREMAST LISTOPT/YES/
$     ENDJOB
```

To add, delete, or modify a file in the permanent file system, use the following deck setup:

```
$     IDENT
$     FILSYS

      file system control cards

$     ENDJOB
```

The first of the file system control cards must be:

USERID PERM$PERM/PERM

To list the contents of the perm files:

CLIST PERM$PERM

To add a new file use:

FCREAT PERM/name,MODE/mod/,SIZE/m,n/

4.18 SYSTEMS INFORMATION BULLETIN #22 (Cont.)

where: name = name of file
mod = SEQ or RAND
m = initial size of file) usually the same
n = final size of file)

To delete an old file use:

FPURGE PERM/name

where: name = name of file

See the FILE SYSTEM manual for other potions and other file system control cards which could be used.

Once the file is created, the object decks can be put on the file in the same manner as GECOS-II by using BMC.

4.19 SYSTEMS INFORMATION BULLETIN #23 4/18/69

SUBJECT: Accounting Cards for GECOS-III FROM: IN-DAT-41 Truitt

GECOS-III has been modified to punch accounting cards rather than write an accounting tape, when the operator types ACCNT. All typewriter messages from the system remain as described in GECOS-III SDL-1 documentation, except that references to an accounting tape have been deleted. The accounting cards are punched using SYSOUT, so the cards may be recovered if the punch fails after the accounting purge is complete.

At present, a slave program must be run after editing and perm restore are complete, to make this modification to the accounting procedure. A new Total Systems Tape containing this modification will be created at a later date.

4.20 SYSTEMS INFORMATION BULLETIN #24 4/25/69

SUBJECT: GE-635 Peripheral Equipment: DSU-270 FROM: IN-DAT-41

The DSU-200 disc units are being replaced by DSU-270 disc units. A comparison of the two disc units under GECOS-III is shown in the following table:

C3

4.20 SYSTEMS INFORMATION BULLETIN #24 (Cont.)

	DSU-200	DSU-270
Average Access Time	199 ms	19 ms
Total Number of *Links	2048 links	3333 links
Number of Links Available to Users	890 links	1680 links x 3840 words
Data Block Size	40 words	64 words

*Each link contains 3840 36-bit words.

All programs using GEFRC to read or write on disc will not be affected. This includes those programs coded in FORTRAN or COBOL. Programs which use GEINOS to process linked files on the disc will also be unaffected. However, those programs which which use GEINOS to process random disc files must be altered since the data block size has increased from 40 words to 64 words. Routines which calculate the relative block address for the Seek Disc Address command will have to be modified to use 64 words as the data block size.

With GECOS-III, 300 links of drum storage are presently available to the user. The DSU-270 access time now compares favorably with the 17 ms average access time of the drums. Users are urged to use disc rather than drum storage due to the larger amount of disc storage available.

Any user who plans to approach the total amount of disc or drum storage avilable should consult IN-DAT-41. Users storage available may be decreased slightly in the future by system changes.

4.21 SYSTEMS INFORMATION BULLETIN #25 4/25/69

SUBJECT: Conditional Control Card ($ IF) FROM: IN-DAT-41
Truitt

The IF conditional control card described on pages 156-157 of the GECOS-III manual (1518A) functions properly only if each $ IF card is immediately followed by a $ BREAK control card. The BREAK control card is described on page 160 of the same manual.

4.22 SYSTEMS INFORMATION BULLETIN #26 5/9/69

SUBJECT: COBOL/Sort-Merge Information FROM: IN-DAT-41
Truitt/8767

I. Under GECOS-III, it is necessary to define all 80 columns of a card read by a Cobol program. For example, if the program defines only the fields in the first 40 columns, a Working Storage Area of 7 words is set up. However, all 14 words of the card image are moved into the W.S.A. as specified by the Record Control Word. Therefore 7 words following the W.S.A. are overlayed, which may result in a program abort.

II. The Sort/Merge software will not function correctly if the FILCB macro specifies double buffering. Therefore single buffering must be specified by setting the first buffer address to one and leaving the second buffer address null. See page 22 of the Sort/Merge manual, CPB-1005C. G.E. is aware of this problem and will send us a correction in 4 to 6 weeks.

4.23 SYSTEMS INFORMATION BULLETIN #27 5/9/69

SUBJECT: Device Commands FROM: IN-DAT-41/Truitt

In the past, the device command pseudo-ops described in Chapter 11 of the GECOS-III manual did not assemble correctly. Therefore, many old programs set up these commands using the OCT pseudo-op.

Example:

```
MME     GEINOS

OCT     550000000000
 ,
 ,
 ,
```

would be used instead of

```
MME     GEINOS

WEF
 ,
 ,
 ,
```

to write an EOF on tape. However, when the GMAP assembler was corrected to assemble device command pseudo-ops, it assembled them differently.

4.23 SYSTEMS INFORMATION BULLETIN #27 (Cont.)

For example, the WEF command is assembled as 55 0000 020001 rather than 550000000000. GECOS-II would accept either version, but GECOS-III will not recognize 550000000000 as a legal device command. Therefore it is strongly recommended that all programs using MME GEINOS be reviewed and those using the OCT pseudo-op to set up device commands be re-assembled with the proper device command pseudo-ops.

4.24 SYSTEMS INFORMATION BULLETIN #28 5/9/69

SUBJECT: GE-635 Control Cards under GECOS-III FROM: IN-DAT-41
Truitt/8767

The following three control cards have been modified in GECOS-III.

I. $ INCODE (CPB-1518A, page 116) now has a fifth option, "GE635," which specifies the GE-635 character set. This card can be used to stop character transliteration specified by a previous INCODE control card.

EXAMPLE:

```
$ INCODE    IBMC

     '  BCD Input Cards
     '
     '
$ INCODE    GE635

     '  BCD Input Cards
     '
     '
```

Since this option was added on site, it will not be described in any G.E. manual.

II. In BMC, $ CONVRT is no longer a valid control card. $ CONVER must be used instead. The software may be modified to accept either control and in the near future.

III. In UTILITY, the option LINE/n on the $ QUTIL card has not been implemented in GECOS-III at the present time.

Two new File Control cards are described on page 56 of the GECOS-III manual (CPB-1518A). Systems programming strongly recommends the use of the $ FILE card for scratch files to utilize the system more effectively. Note that the $ MASS card will not work on System 1 until the DSU 270's are installed on the system.

4.25 SYSTEMS INFORMATION BULLETIN #29 8/28/69

SUBJECT: Overlaying the Slave Prefix FROM: IN-DAT-41

A slave program which attempts to read input data into its slave prefix (the first 100_8 cells of the program) will be aborted. This modification has benn made to GECOS III to prevent master mode aobrts caused by such overlays. In the past, input data has been read into the slave prefix in the following cases:

(1) The input buffer was undefined.
(2) The same buffer was defined for input and output where GEFRC subroutines were used for I/O. This programming technique should not be used, since GEFRC uses the buffer control word as a DCW for data transfer.

4.26 SYSTEMS INFORMATION BULLETIN #30 9/3/69

SUBJECT: Capability of Mass Storage Devices on each GE-635 System FROM: IN-DAT-41

	Drum	MSS	DISC(270)
Average Access Time (Milliseconds)	18	20	26
Total Storage (Links) (1 link = 3840 words)	408	1708	3333
Maximum Storage Allocatable to One Slave File (Links)	300	--	726
Minimum Storage Allocatable to One Slave File (Links)	1	--	1
Effective Transfer Rate Rate (1000 words/sec,)	61.6	102.4	33.3* 43.5 50.1

* This rate depends on which zone is being accessed. (See DSU 270 manual CPB-1419).

NOTE: Only one set of MSS units exists at present and must be switched to the required system. At the present time MSS storage capability is only available to real-time programs. In the near future, this storage will be available for nonreal-time programs also. Details concerning usage will be issued at that time. IN-DAT-41 will provide assistance and information to anyone, who in the meantime, would like to plan for this usage in the future.

4.27 SYSTEMS INFORMATION BULLETIN #31 10/17/69

SUBJECT: Named Device Capability Under GECOS III

FROM: IN-DAT-41

NOTE: This bulletin updates and replaces Systems Information bulletin #21.

One of the features recently developed for GECOS-II and further refined in GECOS-III is names devices. This feature should be used by operations to efficiently allocated printers for jobs requiring multi-part paper and other special forms. Named devices should also be used by real-time jobs to specify the number of tape controllers required and the desired location of magnetic tapes. Name device must be used, under GECOS III , for real-time jobs to obtain three or more tape controllers as required to record 5K or 6K measurements.

To use a device by name (see accompanying table for device names) the device should be dedicated. This can be done by the console operator. Once dedicated the device can be allocated only by name until the console operator undedicates it.

The use of named devices can eliminate the bothersome task of having to release and re-assign printers.

An example of named devices would best illustrate their use.

```
$    SNUMB    RT6
$    IDENT    xxxxx.....xxxxx
$    SELECT   504A2
$    EXECUTE DUMP, ON4
$    LIMITS   999,121000,,
$    TAPE     T1,TC0D,,XXXX,,OUT-T-1
$    TAPE     T2,TC1D,,XXXX,,OUT-T-2
$    TAPE     T3,TC2D,,XXXX,,OUT-T-3
$    TAPE     T4,TC3D,,XXXX,,OUT-T-4
$    TAPE     W1,TD4D,,XXXX,,OUT-W-1
$    TAPE     W2,TD5D,,XXXX,,OUT-W-2
$    TAPE     W3,TD6D,,XXXX,,OUT-W-3
$    TAPE     W4,TD7D,,XXXX,,OUT-W-4
$    TAPE     T8,TA0D,,3477,,PIPD
$    TAPE     T5,TA1D,,0632,,INIT
$    TAPE     S2,TA2D,,XXXX,,GUIDANCE-S-2
$    TAPE     S1,TA3D,,XXXX,,GUIDANCE-S-L
$    PRINT  PR,SF3
$    PRINT  RP,SF4
$    ENJOB
***  EOF
```

4.27 SYSTEMS INFORMATION BULLETIN #31 (Cont.)

With this set up the real-time job would get a printer on channels 13 and 14, 4 tapes on channels 2 and 4, and 8 tapes on channels 3 and 5. In addition the tapes on channels 3 and 5 will be crossbarred and in effect the real-time job would be able to use tape controllers 2, 3, and 5. Note that when only one tape controller is required for output by a real-time job, it can be specified by named devices, leaving three controllers free for batch work.

The crossbarring of tapes for real-time jobs is keyed off of the second character of the tape name, with letter A associated with channel number 2, B with 4, C with 3, and D with 5. It is recommended that real-time jobs request only those tape controllers required for processing, as no batch job can share a controller specified by a real-time job. In particular, it is recommended that real-time jobs request the tape controller named A, C, or D if only one controller is required. If two controllers are required, use A together with either C or D. If three controlllers are required, use A, C, and D.

All tape handlers not used by the real-time job will be available for batch work, unless the real-time job requests three controllers. Then only the unused handlers on pubs 2-4 are available for batch work.

NAMED DEVICE TABLE

PRINTERS	CHANNEL	UNIT
SF1	7	
SF2	10	
SF3	13	
SF4	14	
READERS		
CR1	6	
CR2	11	
TAPES		
TA0,TB0	2	0
TA1,TB1	2	1
TA2,TB2	2	2
TA3,TB3	2	3
TA4,TB4	2	4
TA5,TB5	2	5
TA6,TB6	2	6
TA7,TB7	2	7
TC0,TD0	5	0
TC1,TD1	5	1

4.27 SYSTEMS INFORMATION BULLETIN #31 (Cont.)

NAMED DEVICE TABLE (Cont.)

PRINTERS	CHANNEL	UNIT
TAPES (Cont.)		
TC2,TD2	5	2
TC3,TD3	5	3
TC4,TD4	5	4
TC5,TD5	5	5
TC6,TD6	5	6
TC7,TD7	5	7
PUNCH - NOT NAMED		
PPT - NOT NAMED		
TYPEWRITERS		
TY1,TY2,TY3,	8	(NOTE: The typewriter names are for operator use
TY4	12	only and should not be used in control cards.)

4.28 SYSTEMS INFORMATION BULLETIN #32 10/20/69

SUBJECT: Processor Time on GE-635 Limits Card FROM: IN-DAT-41

The amount of processor time indicated on $ LIMITS cards should be carefully examined for accuracy. Large amounts of computer time can be wasted when an unreasonably amount of time is indicated and the program gets hung in a loop.

A large job with several activities was recently aborted because 999 was indicated in the LIMITS card of each activity. This total amount exceeded the maximum of 140 processor hours presently allowable by the system.

4.29 SYSTEMS INFORMATION BULLETIN #33 11/25/69

SUBJECT: SDL 1.3.4

The following changes will be implemented with SDL 1.3.4.

1. Operator input error button:

   When the operator hits the Operator Input Error button on console input, both GEPR and GSEC (for non-real-time and real-time programs) automatically re-issue the read command to give the operator another chance. Under SDL 1.3.3 neither module clears the slave program's input buffer, so that whatever the operator typed in first will still be in the buffer unless it is completely overlaid by what he types in on the re-try. The programmer can check the DCW residue to see exactly how many characters the operator typed in. Under SDL 1.3.4 the real-time program buffer will be zeroed out before the read command is re-issued by GSEC.

   On other errors in typewriter I/O, both GEPR and GSEC will return the bad status to the user. Therefore it is recommended that both real-time and non-real-time programmers check the status return words.

2. Real-time slave abort in courtesy call:

   If a real-time aborts while in a courtesy call, the program will be terminated immediately rather than waiting til a relinquish at main level, which sometimes caused a system hangup.

3. MME RTMEAS:

   The real-time program can now set the number of measurements (3K, 5K, or 6K) and override the number specified on the $ SNUMB card by using MME RTMEAS (400014001000) and having a value of 3, 5, or 6 in the Q-register. Any value other than one of these will cause the program to abort. A console message will reflect any change in the number of measurements.

4. MME RTCRIT:

   The real-time program can reset the mode from critical to non-critical by using MME RTCRIT (400002001000). If the Q-register is equal to zero, the mode is set to non-critical. If it is non-zero, the mode is set to critical. A console message will reflect the change in mode.

4.29 SYSTEMS INFORMATION BULLETIN #33 (Cont.)

5. MSS:

   Both real-time and non-real-time programs can now use the Mass Storage Subsystem. More information will follow.

6. Real-time tape release:

   When a tape is released in real-time, the tape will be unloaded, unless it has a release disposition code.

7. Tape dismounting:

   Jobs which dismount more than 13 tapes will no longer cause a system abort. Sixteen tapes can now be dismounted.

8. LENTRY:

   Fortran programs can now call entry point LENTRY of the subroutine LINK.

9. GMAP Label Table:

   The GMAP Label Table has been expanded by 512 additional locations.

10. Real-time I/O queue overflow:

    An R3 real-time slave abort no longer occurs when the real-time I/O queue is overflowed by requests on the display write channel. Instead, one entry in the queue is deleted and the following message is typed out:

    RT-REQ. DELETED ON CH-6.

11. The hardware has been modified so that a DU or DL modification on a store instruction will result in a slave F8 abort.

4.30 SYSTEMS INFORMATION BULLETIN #34 2/17/70

SUBJECT: Slave program recovery after a lost interrupt on the Display Data Channel

FROM: IN-DAT-41/ Celsor

The GE-635 software has been modified so that the real-time slave program may detect a lost interrupt and initiate any recovery action desired.

The real-time slave program can determine the elapsed time from a given connect on the Display Data channel. The status word associated with the given Display Data request will have bit-1 set to 1, when the connect has been issued. Upon detecing bit one set to 1, the program may use the timer courtesy call to determine if the elapsed time from the time of the connect exceeds the anticipated transfer time of the data on the Display Data channel. If the elapsed time does exceed the transfer time, the real-time slave program must issue a MME RTFAKE (BOOL 400030) and then reissue the request in order to recover. This reissued request will be the latest entry in the I/O queue for the Display Data Channel. Connects are issued for each entry in this queue in the order in which they are placed in the queue.

4.31 SYSTEMS INFORMATION BULLETIN #35 2/17/70

SUBJECT: Using the largest DBA to determine Flag Table Scan

FROM: IN-DAT-41/ Johnson

Presently the Ge-635 software scans a 32 word DBA flag table to determine which data blocks have been received between timer interrupts.

A more efficient scan method is now available. To accomplish this, the real-time slave program has been provided a MME so that the program can tell the system the largest DBA that is required during the run. The system will then terminate the scan at the largest DBA required.

This MME is a MME RTDBA (BOOL 400026). It should be issued just after the MME RTMODE. If the RTDBA is not used, the original 32 word search will remain in effect.

The largest DBA required should follow the MME (See example below) and be binary and right adjusted. Values smaller than 1 or larger than 32 (decimal) will cause a "RD" slave abort. It is suggested that the largest DBA number be set initially at 32 (OCT 40) so failure to override this value will not change the system or abort the slave.

4.31 SYSTEMS INFORMATION BULLETIN #35 (Cont.)

To override the OCTAL 40 simply store the required value into location of the MME plus one of the program. The highest DBA # required can be calculated internally or read in on a data card, as the programmer desires.

EXAMPLE: RTDBA

```
        MME RTMODE
L       MME RTDBA (BOOL 400026)
L+1     OCT 40 (Location for highest DBA # initially set
          at 32)
```

4.32 SYSTEMS INFORMATION BULLETIN #36 3/13/70

SUBJECT: Sequence check on Object Decks

The GECOS module GEIN does not automatically check sequence numbers on object decks as they are read in. To obtain sequence checking, the option CKSEQ must be called for on the $ OBJECT control card.

```
          1  8        16
EXAMPLE:  $  OBJECT   CKSEQ
```

If this option is used and the object deck is out of sequence, the job will be deleted.

When the program is reassembled, the CKSEQ option will be punched into the $ OBJECT card if it appears in the second field of an LBL instruction in the symbolic deck.

```
          8      16
EXAMPLE:  LBL    ,CKSEQ
```

4.33 SYSTEMS INFORMATION BULLETIN #36 4/3/70

SUBJECT: Illegal use of MME's in courtesy calls

The use of illegal MME's in courtesy calls is causing serious system malfunctions. The Comprehensive Operating Supervisor manual (CPB-1518A) indicates in the description of each MME if that MME should not be used in a courtesy call. Care must also be taken in the use of GEFRC routines in courtesy calls, since some of these routines use these MME's. The GEFRC manual (CPB-1003F) states in the description of each routine

4.33 SYSTEMS INFORMATION BULLETIN #37 (Cont.)

when it cannot be used in courtesy calls. The system will be modified in the near furture so that any batch program which attempts to use an illegal MME in a courtesy call will be aborted.

4.34 SYSTEMS INFORMATION BULLETIN #38 4/24/70

SUBJECT: Modifications to GEPR

The following modifications will be made to the I/O error recovery module, GEPR, on the next GE-635 systems update.

1. During error recovery on input tapes, GEPR now passes 85 feet of blank tape before returning a blank tape status to the program. GEPR will be modified so that only 25 feet of blank tape will be passed. During the investigation of this problem, it was found that the GECOS III manual (CPB-1518A) is in error on page 331. Only 30 inches of tape are passed with each read command, not 120 inches.

2. If a binary "77" EOF mark is encountered, GEPR now terminates the user program with an "EP" abort. The status is not returned to the program, and the operator is not given an option to attempt to correct the error. With the next update GEPR will be modified to return the status to the program. This correction will make the action taken by GEPR consistent with the write-up on page 335 of the GECOS III manual.

4.35 SYSTEMS INFORMATION BULLETIN #39 4/28/70

SUBJECT: Change in MME RTFAKE

The MME RTFAKE which is presently used to simulate an interrupt on the display-data channel will be modified in SDL 1.3.11 to also permit simulation of an interrupt on the MSS. With added capability, the real-time program can allow the MSS to be switched from one system to another with I/O still queued up, and without causing a hangup or an R3 abort. It also gives the real-time program the capability of determining if the MSS is connected to the desired system prior to use, and allows operator intervention if necessary to correct the switching of the MSS without terming and reloading the real-time program.

4.35 SYSTEMS INFORMATION BULLETIN #39 (Cont.)

OLD Select Sequence

```
L     MME RTFAKE  (BOOL 400030)
L+1   Return
```

Result: An interrupt was simulated on the display-data channel.

NEW Select Sequence

```
L     MME RTFAKE  (BOOL 400030)
L+1   ZERO FC pointer, 0
L+2   Return
```

File Codes Allowed:

```
R6    Display-data  (BCI 1,0000R6)
MS    MSS           (BCI 1,0000MS)
```

Result: An interrupt will be simulated on the channel specified by the file code.

Restrictions:

1. MME RTFAKE is only allowed on the display-data channel and the MSS.
2. This MME is not allowed in a courtesy call.

Example:

```
MME   RTFAKE
ZERO  FC, 0
      .
      .
      .
FC    BCI  1,0000MS
```

4.36 SYSTEMS INFORMATION BULLETIN #40 5/28/70

SUBJECT: Labels used as SYMDEF's by the User's Library and the Subroutine Library

If a batch programmer defines a label as a SYMDEF and the same label is also used as a primary or secondary SYMDEF in a library subroutine, the loader will not search the libraries for that subroutine but will reference the programmer's label. To avoid this error, no label with a dot (.) in it should be used by a batch program as a SYMDEF or SYMREF. Also the labels listed in Attachments 1 and 2 should never be used as SYMDEF or SYMREF symbols.

4.36 SYSTEMS INFORMATION BULLETIN #40 (Cont.)

Subroutine Library SYMDEF's to be avoided by Programmers

ALOG	DMOD	FXEM	PUTBK	XXEND
ALOG10	DSIN	FXFDV *	PUTSZ	X1060
ANYERR	DSORT	FXOPT	RDREC	X1091
ATAN	DUMP	GET	READ	X1095
ATAN2	DVCHK	GETBK	RELSE	X4000
BEGIN	EDATE	IDLINK	REWND	15AUG5
BSREC	EPRINT	IOEDIT	SETBUF	
BSTFM	ERRLK *	IOPO21	SETFCB	
CABS	ETIME	IOPO24	SETIN	
CCOS	EXIT	LENTRY	SETLGT	
CEXP	EXIT	LINK	SETOUT	
CLOG	EXP	LLINK	SIN	
CLOSE	FCLOSE	OPEN	SLITE	
COPY	FICNVD	OVERFL	SLITET	
CORECT	FICNVS	PART	SORT	
COS	FLGEOF	PDUMP	SSWTCH	
CSIN	FLGERR	PRINT	STORE	
CSORT	FOCNVD	PRTO02	TANH	
DATAN	FOCNVS	PRTO04	TDS	
DATAN2	FORCE	PRTO24	WAIT	
DCOS	FSREC	PRTO31	WEF	
DEBUG	FSTFM	PRTO32	WRITE	
DEBUG	FXALT	PRTO35	WTREC	
DEXP	FXCODE *	PRTO51	XCMNT	
DLOG	FXDV *	PUNCH	XDUMP	
DLOG10	FXDVCK	PUT	XTMIII	

(Symbols followed by an asterisk are secondary SYMDEF's)

4.36 SYSTEMS INFORMATION BULLETIN #40 (Cont.)

User Library SYMDEF's be avoided by Programmers

APLOTV	CWIDE	FTF	LABLV	PLOTFT	TABL3V
APRNTV	DARCOS	FTX	LINEV	PLOTV	TABL5V
ARCOS	DARSIN	GRID1V	LINRV	PLOUTV	TABL6V
ARSIN	DATER	GYAXIV	LVH	POINTV	TABL7V
ATANH	DATE1	HALT1	LVW	PRFCB	TABL8V
BCD	DATE2	HOLBIN	MGRIDV	PRINTV	TABL9V
BDCERS	DATE3	HOLDIV	MLINV	PRT	TAB11V
BDCSV	DATE4	HOLDOV	MODATE	RESETV	TAB15V
BHSAVE	DATE5	HOLD1V	MTB	RITE2V	TPW
BIN	DATE6	HOLIND	MTL	RITSTV	UVO1O
BINARY	DBUG1	HOLLV	MTR	SCERRV	UXV
BINHOL	DBUG1	HTH	MTT	SCLSAV	UYV
BNBCDV	DENSW	IBM	NOFRV	SCTAB	UYO1O
BRITEV	DFLBTD	IDV	NVO1O	SERREV	VCHARV
BRO1O	DFLDTB	IGCH	NWDSV	SERSAV	VTAB
BUFV	DFXBTD	INTBTD	NXV	SETCIV	WDCNT
BUTTY	DFXDTB	INTDTB	NYV	SETCOV	XSCALV
CAMRAV	DMP	IN2	NYO1O	SETMIV	XSO1O
CHARV	DXDYV	IROW	OBUFV	SETMOV	XTX
CHITE	EDATE	ISPACE	OPEN *	SFLBTD	YRDAY
CHSETV	ERRORX	ISTCH	OPEN *	SFLDTB	YSCALY
CHSIZV	FAINTV	ITI	OUT	SFXBTD	YSO1O
CKID	FAO1O	IXV	OUTPUT	SFXDTB	
CLEAN	FORMV	IXO1O	PLCNTV	STOPTV	
CLUSEM	FRAMEV	IYV	PLFCB	TABL1V	
CSEL	FRCT	IYO1O	PLID	TABL2V	

(Symbols followed by an asterisk are secondary SYMDEF's)

4.37 SYSTEMS INFORMATION BULLETIN #41 5/28/70

SUBJECT: SHARING MEMORY WITH GELOAD

When a program is loaded by GELOAD, an additional 9K of memory is allocated to the user at the lower end of his core. GELOAD is loaded into this 9K area. It then proceeds to load the user's object decks into his area. When GELOAD has finished its task it releases its memory.

It is possible for the user to share memory with GELOAD, thus reducing the amount of core needed to get his program into execution. The shared memory must be at the lower end of the user's core and must not be used during loading. The following types of core use fit these requirements: blank or unlabeled common; block or labeled common that is not initialized by a BLOCK DATA subprogram (this also requires the LOCOMN option on the $ OPTION card); buffer space used by FORTRAN files; and unused memory. Obviously, the $ LOWLOAD card may not be used if the user is going to share memory with GELOAD.

The user can share up to 9K in increments of 1K of core with GELOAD. The amount of core to be shared is specified by the 3rd field in the $ LIMITS card. This shared memory will not be released by GELOAD.

USING H* FILES TO BYPASS GELOAD

If the user has a production program, it is possible to create an H* file and thus bypass GELOAD on subsequent runs. There are two advantages to using and H* file: (1) faster allocation, since the additional 9K of core for GELOAD is not needed; (2) faster loading, since the only thing that needs to be done is to read the program from the H* file directly into core.

To create the H* file, the user should use the SAVE/XXXXXX option on the $ OPTION card. The user must also have an H* file. This H* file can be either a tape or a perm file. The user may create his H* file and let the program continue into execution or he may use the NOGO option and build his H* file then stop execution. If the user has GELOAD build his File Control Blocks, then all of his files (even optional files) must be defined. If the NOGO option is used, then the files may be defined by $ FFILE cards only. This applies only to files for which GELOAD creates File Control Blocks.

4.37 SYSTEMS INFORMATION BULLETIN #41 (Cont.)

Once the H* file has been built, the program may be reloaded by using the program KSCHS with a one card data file with file code KS. This one data card contains in columns 1-6 the name used by the SAVE option. It must also contain FCB in columns 16-18 if the user had GELOAD build any File Control Blocks (i.e., he used either the FCB option or the FORTRAN option on the $ OPTION card).

EXAMPLE NO. 1 - CREATE H*

```
$       OPTION    FORTRAN, NOGO, SAVE/BALLON
$       SELECT    RWDV6
$       EXECUTE
$       LIMITS    05,36K,8K
$       TAPE      H*,A1D,,XXXX,,H*-TESTER
$       FFILE     01,NOSRLS,FIXLNG/40,NSTDLB,ERRXIT/TAPERR,LODENS
$       FFILE     03,STDLBL
$       ENDJOB
```

EXAMPLE NO. 2 - USE H*

```
$       PROGRAM   KSCHS
$       LIMITS    20,36K
$       TAPE      H*,A1D,,XXXX,,H*-TESTER
$       TAPE      01,A2D,,XXXX,,WIND-MON
$       TAPE      HU,A3D,,XXXX,,HUNTSVILLE
$       TAPE      HO,A4D,,XXXX,,HOUSTON-200BPI
$       PRINT     03,A5R
$       INCODE    IBMF
RWDV6
000000.00   700317 HUSTON HUNTSV FPS   SATURN 090.00       004
025.000M  0.00157077  0041.1480822  0.00261795  0091.4401828  SHEARS
APOLLO 13         LAUNCH
$       DATA      KS
BALLON            FCB
$       ENDJOB
```

NOTES:

1. KS abort if KS file not present.

2. $ TAPE H* could be replaced by
   $ PERM H*

3. When H* is created, either source or object decks may be used.

4. The memory limits used to create the H* and to use the H* must be the same.

5. Any I* data should preceed the $ DATA KS card.

4.38 SYSTEMS INFORMATION BULLETIN #42 9/16/70

SUBJECT: Producing a GE-635 Source Deck from a COMDK

The following file edit run can be used to produce a GE-635 Source Deck from a COMDK:

```
1     8          16                        73
$     IDENT
$     FILEDIT    SOURCE, NOBJECT
$     DISC       K*,A8R, 5L
$     SYSOUT     *7
$     DATA       M*,, COPY
$     GMAP                                 NAME1
      (COMDK)
$     GMAP                                 NAME2
      (COMDK)
$     GMAP
        .
        .
        .
$     ENDEDIT
$     ENDCOPY
$     DATA       *C,, COPY
$     PUNCH      NAME1, BCD                NAMES MUST BE
                                           SPECIFIED IN SAME
$     PUNCH      NAME2, BCD                ORDER AS THEY
                                           APPEAR ABOVE
        .
        .
        .
$     ENDEDIT
$     ENDCOPY
$     ENDJOB
***EOF
```

4.38 SYSTEMS INFORMATION BULLETIN #42 (Cont.)

NOTE 1: This output deck is in GE-635 character set. However, the above file edit run can be altered to produce a deck in IBMF format as follows:

1. Change the $ SYSOUT *7 card to a $ TAPE *7, A4S card. (The save disposition code is required.)
2. Insert the following deck, which is a second activity, in front of the $ ENDJOB card:

```
1    8          16
$    CONVER
$    OUTPUT     IBMF
$    TAPE       IN, A4R
$    PUNCH      OT, A5R
```

Other transliterations available are IBMC & GE-225.

NOTE 2: IBMC and GE-225 format decks can be obtained by substituting IBMC or GE-225 for IBMF above.

4.39 SYSTEMS INFORMATION BULLETIN #43 10/9/70

SUBJECT: Load Table and Program Overlap Under SDL 3

The loader has been expanded under SDL 3. There is now less room for the load tables, and some programs may terminate with an L3 abort due to program and load table overlap. This problem may be corrected by specifying a negative value in the third field of the LIMITS control card, so that this value will be added to the size of the loader. (See CPB-1518B, page 44). It is anticipated that this problem will affect relatively few programs. The value -2000 has been the maximum value required thus far to correct this problem.

Example:

```
$  LIMITS   30,40000,-2000.10000
```

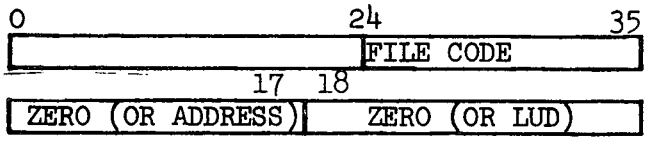
With this LIMITS card, the amount of core allotted to the loader will be 11,000 locations rather than the standard 9,000 locations.

4.40 SYSTEMS INFORMATION BULLETIN #44 10/30/70

SUBJECT: Use of The Real-Time MME RTMORE to Obtain an Additional Tape Handler

ENTRY: At the time of execution of this MME the A-register must contain the File Code in bits 24-35. If a reel number is to be typed with the mounting instructions, the address of the reel number should be in the Q-register, bits 0-17. (Reel number format: nnnnn Left justified). If a named device is desired the LUD must be in the Q-register, bits 18-35.

CALLING SEQUENCE:



L	MME RTMORE
L+1	ZERO A,B
Z+2	Denial Return
L+3	Successful Return
A =	1 for 7- track tape
	3 for 9- track tape
B =	0 if a scratch is wanted
	1 if a tape is to be mounted

TYPEWRITER MESSAGES:

* MNT TAPE S#sssss i-cc-dd REEL # RMORE
  (if C(Q) 0-17 is zero)

* MNT TAPE S#sssss i-cc-dd REEL #nnnnn
  (if C(Q) 0-17 contains the address of a reel no.)

4.40 SYSTEMS INFORMATION BULLETIN #44 (Cont.)

*RDY TAPE S#sssss i-cc-dd REEL # RMORE

*LBL TAPE S#sssss i-cc-dd REEL # RMORE
for scratch tapes.

RESTRICTIONS:

The MME RTMORE may not be used during courtesy call. User must be prepared for denial.

4.41 SYSTEMS INFORMATION BULLETIN #45 11/31/70

SUBJECT: Use of the Real-Time MME RTMØRE to Obtain Additional Memory

Calling Sequence:

```
L     MME   RTMØRE
L+1   ZERØ  A,B
L+2   TRA   D
L+3   ZERØ  CC
L+4   NULL
```

A must be zero

B is the number of 1024 word blocks of additional memory requested

D is the address of the denial return

CC is a courtesy call address

If the request can be satisfied, control will be returned to location L+4, where the real time program should do a MME RTRELC. Since it will usually be necessary to either move or swap a GECØS slave program, the real time program must relinquish control before the request can be satisfied. Once the necessary memory has been allocated to the real time program, the courtesy call whose address is given in location L+3 will be serviced. The real time program must not assume that it has the additional memory until the courtesy call has been serviced. It is not necessary for the MME RTRELC to immediately follow the MME RTMØRE, but GECØS must get control before the request can be satisfied. In the event that GECØS has aborted, the additional memory will be given to the real time program and the courtesy call will be serviced immediately.

4.41 SYSTEMS INFORMATION BULLETIN #45 (Cont.)

The request for additional memory will not be denied unless a previous request has not yet been completed or unless the request is larger than the amount of unused memory plus the amount assigned to GECØS slave jobs.

4.42 SYSTEMS INFORMATION BULLETIN #46 11/5/70

SUBJECT: H* Files

Under SDL 1, the H* file was assigned to the disc if no control card was used to specify otherwise. Under SDL 3, the H* file is assigned to the drum. An N8 abort will occur if there are not enough available links on the drum for this file. Programmers can force H* to the disc by defining it with a $ DISC or a $ MASS control card. The system will be modified at a later date so that H* is assigned to the disc.

4.43 SYSTEMS INFORMATION BULLETIN #47 11/5/70

SUBJECT: Batch Programs Typewriter Messages and COMMENTS Cards.

Programmers who use the console typewriter should observe the following programming conventions:

1. All messages should be preceded by the SNUMB of the program. The SNUMB, in BCD format, may be found in word 30 (decimal) of the slave prefix.

2. If the operator is to reply to the typewriter message, the proper response whould be given in the question.

Example:

S # 12345 TYPE YES IF CARDS ARE TO BE PUNCHED

3. If a message concerning a tape is typed, the IØC number, channel number, and device number of the tape should be given. If GEFRC is used, these may be found in LOCSYM-1 of the File Control Block. If GEINØS is used, the programmer should use MME GEFADD.

Example:

S #12345 MNT INIT REEL ON 0-05-01

4.43 SYSTEMS INFORMATION BULLETIN #47 (Cont.)

4. Typewriter messages from batch programs should be concise and should conform to systems message formats as much as possible.

COMMENTS Cards:

Under SDL 3, the SNUMB of the programs will precede the COMMENT on the typewriter. The SNUMB is added by the system; this will shorten the number of characters available on the COMMENT card by 6.

4.44 SYSTEMS INFORMATION BULLETIN #48 11/23/70

SUBJECT: Additional Error Checks Under SDL-3

Under SDL-3 several additional programming errors, previously warned against, now result in program aborts. The most notable of these is with variable Fortran formats (formats constructed or read in at object time). The Fortran manual has stated that these formats begin with a left parenthesis since at least December 1966. However, in the past, the program was not aborted. Under SDL-3 it is.

Programmers should be familiar with programming rules as stated in the manuals. A strong effort is being made by the Vender to update the software to abort all jobs not following manual restrictions. This effort is to assure correct results, and alert programmers to error conditions.

4.45 SYSTEMS INFORMATION BULLETIN #49 21/9/70

SUBJECT: Conversion of IBM-360 COBOL Programs to GE-635

IN-DAT-41 has received a copy of a Honeywell application program that should be of assistance in converting a program written in IBM-360 COBOL to GE-635 COBOL. This program accepts IBM source card input from cards or tape and outputs a GE-635 format deck.

It is doubtful that this program represents a utopian conversion process for all IBM-360 COBOL programs. This program will hopefully prove of assistance, however.

4.45 SYSTEMS INFORMATION BULLETIN #49 (Cont.)

A copy of the program binary deck and documentation may be checked out from IN-DAT-41. Should this program prove of value to a sufficient number of users, the program can be placed in the PERM FILE Library and documentation made available for using organizations.

4.46 SYSTEMS INFORMATION BULLETIN #50 1/29/71

SUBJECT: GE-635 Keypunch Characters

The attached chart gives the GE-635 octal representation and character printed when each card punch is processed by the IBMF, IBMC, and IBMEL transliterations. The results with no transliteration are also shown.

NOTE: The characters on all coding sheets given to the keypunch operations will be assumed to be IBMF characters and be punched with the corresponding punches. Any deviations from this must be specifically requested.

In the IBMF and the IBMC transliteration five characters can be punched with two different codes to yield the same octal code and print character. The underlined character and code will be the one normally punched by the operator.

USE OF SPECIAL CHARACTERS ON GE-635

ARD UNCH	RESULT OF IBM TRANSLITERATION OCTAL CODE	PRINT CHAR.	RESULT OF NO TRANSLITERATION (GE) OCTAL CODE	PRINT CHAR.	RESULT OF IBMEL TRANSLITERATION OCTAL CODE	PRINT CHAR.	RESULT OF IBMC TRANSLITERATION OCTAL CODE	PRINT CHAR.
?-8	12	[	12	[	15	:	12	[
-8	75	=	13	#	13	#	75	=
-8	57	'	14	@	14	@	76	"
-8	13	#	15	:	57	'	13	#
6-8	16	>	16	>	75	=	16	>
7-8	17	?	17	?	76	"	17	?
12	60	+	32	&	32	&	60	+
12-3-8	33	.	33	.	33	.	33	.
12-4-8	55	]	34	]	36	<	55	]
12-5-8	35	(	35	(	35	(	35	(
12-6-8	36	<	36	<	60	+	36	<
12-7-8	37	\	37	\	37	/	37	\
11-0	40	†	40	†	40	†	40	†
11	52	-	52	-	52	-	52	-
11-3-8	53	$	53	$	53	$	53	$
11-4-8	54	*	54	*	54	*	54	*
11-5-8	55	)	55	)	55	)	55	)
11-6-8	56	;	56	;	56	;	56	;
11-7-8	57	¬	57	¬	20		57	¬
12-0	60	+	60	+	60	+	60	+
0-1	61	/	61	/	61	/	61	/
0-2-8	72	←	72	←	20		72	←
0-3-8	73	,	73	,	73	,	73	,
0-4-8	35	(	74	%	74	%	35	(
0-5-8	75	=	75	=	20		75	=
0-6-8	76	"	76	"	16	>	76	"
0-7-8	77	!	77	!	17	?	12	[

4.47 SYSTEMS INFORMATION BULLETIN #52 3/30/71

SUBJECT: FFILE Card Options for Fixed-Length Tape Records

If the FIXLNG parameter on the FFILE card is set to less than the actual size of the physical record, as described by the BUFSIZ parameter, GEFRC will attempt to deblock the fixed length record.

Example: $ FFILE FC,NSTDLB,NOSRLS,BUFSIZ/300,FIXLNG/20

GEFRC will read 300 words of the record into the buffer and then move 20 words at a time into the Working Storage Area (WSA) of the program. The first time the program reads the tape, the WSA will contain Word 0 through Word 19 of the physical record. The next time a read is issued, the WSA will contain Word 20 through Word 39 of the physical record, etc.

If the programmer wishes to read only the first n words of each physical record, then he must define the BUFSIZ to be the same size as the FIXLNG parameter.

Example: $ FFILE FC,NSTDLB,NOSRLS,BUFSIZ/20, FIXLNG/20

In this case only the first 20 words of each physical record will be read into the buffer area and moved into the WSA.

Note that it is not necessary to allow one word in the BUFSIZ definition for the buffer control word. It has been customary at this installation to define the BUFSIZ as one greater than the FIXLNG parameter. This will not cause an error if the entire physical record is being read and moved to the WSA.

4.48 SYSTEM INFORMATION BULLETIN #53 5/13/71

The System Software will be modified in SDL 3.2.3 to permit recovery from a lost interrupt on the IDC by a real time program. The purpose of this memorandum is to introduce the technique that will be used by the program to initiate recovery.

It is the programs responsibility to run associated checks against elapsed time before requesting interrupt recovery.

4.48 SYSTEMS INFORMATION BULLETIN #53 (Cont.)

The Disc, Drum, and Typewriter channels may not be used. An R5 abort will occur if a request is issued on the Typewriter channel. The request will be ignored if it is issued on a Disc or Drum channel.

The system is designed so that the software queue routine zeroes out both status words of the associated input-output entry (MME RTINOS). The software connect routine places the true channel number in the upper half of status word number two. It also sets the lower half of status word number two non-zero. After an input-output entry has been issued the two status words (SW1 and SW2) may be checked to determine the following:

1. If an input-output entry has been queued but not connected.

   a. SW1 = 0
   b. SW2 = 0

2. If an input-output entry has been queued and connected.

   a. SW1 = 0
   b. SW2 $\neq$ 0

      (1) Bits 0-17; contain the numeric value of the true channel that the connect was issued on.
      (2) Bits 18-35; will always be non-zero.

3. If an interrupt has returned for the associated input-output entry.

   a. SW1 will contain normal status.
   b. SW2 will contain normal status.

A program should use different sets of status words with input-output that is connected on different channels. The software queue, connect, and interrupt processing routines use them to relay status to the program. Faulty status could occur if this technique is not followed.

If the elapsed time from the time of the connect exceeds the anticipated transfer time and SW1 = 0 the recovery procedure may be initiated. The calling sequence that follows should be used.

```
LDQ   SW2
MME   RTFIOC (400005001000)
```

4.48 SYSTEMS INFORMATION BULLETIN #53 (Cont.)

This calling sequence will initiate a true hard ware interrupt on the associated channel. A substatus of all sevens (77) will be returned. This will permit subsequent and outstanding input-output entrys on the associated channel to be processed normally. Timing constraints and resynchronization of input-output on the associated channel is the responsibility of the real time program.

NOTE: A MME RTFIOC could be issued before the data gets transferred and the subsequent arrival of the true interrupt.

1. The MME RTFIOC (a software generated hardware interrupt) would perform like the true interrupt and do the following:

   a. Return a status to the program.
   b. Pay a requested courtesy call.
   c. Delete the first (connected) entry from the input-output queue.
   d. Connect the most outstanding (second) entry in the input-output queue.

2. A channel busy status will indicate the following:

   a. The MME RTFIOC has been issued prior to the transfer of all the data on the associated channel.
   b. A connect had been issued on an outstanding or subsequent input-output entry prior to the transfer of all the data for the initial input-output entry.

3. Status for the initial input-output entry may resemble the following:

   a. May be passed to an outstanding or subsequent input-output entry.
   b. May not be returned because a premature MME RTFIOC generates an additional interrupt and an input-output entry would not exist for it.

4. Improper use of the MME RTFIOC may cause loss of data, and synchronizing problems.

4.49 SYSTEMS INFORMATION BULLETIN #54 6/3/71

SUBJECT: MME GEFADD for the Mass Storage Device

A. Input

The input corresponds to that required for the standard MME GEFADD.

B. Output

1. If a match is found the C(Q) bits 6-11 will contain the device address and the C(A) will be meaningless.

2. If match is not found the output will correspond to that returned for the standard MME GEFADD.

4.50 SYSTEMS INFORMATION BULLETIN #55 7/22/71

SUBJECT: COBOL TRACE

A COBOL Trace program is available on the GE-635. This program will compile the user's program without printing a listing. Next, the user's program will be executed, and each paragraph heading will be displayed as the paragraph is executed. These displays will appear on the same print report as the program's displays. The following control card set-up must be used to execute the COBOL-Trace program:

```
Cols   1    8                16
       ↓
       $    IDENT            .....
       $    SET              3,5
       $    SELECT           COBOL-TRACE
       $    DATA             I*,,COPY
       $    COBOL            COBOL program
                .
                .
                .
                .
       $    ENDCOPY
       $    SOURCE
```

4.50 SYSTEMS INFORMATION BULLETIN #55 (Cont.)

```
$    EXECUTE DUMP        Control cards used by COBOL
                         program during execution
$    LIMITS .....
      .
      .
      .
$    DISC                B*,A5R,25L
$    ENDJOB
***  EOF
```

where ..... indicates variable parameters to be filled in by the programmer.

4.51 SYSTEMS INFORMATION BULLETIN #56 7/29/71

SUBJECT: $ SELECT CARD

Effective August 2, 1971 all GE-635 programs utilizing $ SELECT cards must be altered.

1. The format of the $ SELECT card must be altered as follows:

	(1)	(8)	(16)
Old:	$	SELECT	Name
New:	$	SELECT	KSCPERM/Name

Name if the program to be selected.

2. The following card must be added following the $ IDENT card and proceeding the $ SELECT card:

(1)	(8)	(16)
$	USERID	KSC$KSC

4.52 SYSTEMS INFORMATION BULLETIN #57 8/5/71

SUBJECT: Changes to the FILEDITOR in SDL 3.3

The attached pages contain information on changes and new and extended features of the FILEDITOR. For example, the character transliteration option must be included on the $ MODIFY card only.

$ ENDEDIT

Function

The $ ENDEDIT control card signals the logical end of the Editor Input File (*C). The Source and Object Library Editor uses $ ENDEDIT to control internally the New Source Library (K*) and the Object Library Editor Stack File (B*). These files are multifile and the $ ENDEDIT is used to signal the logical end of the file.

If $ ENDEDIT is the only directive on *C, the entrie contents of the Old Source Library are copied to the New Source Library (all extraneous EOF marks are removed); the entire contents of the Old Object Library are copied to the New Object Library.

Example

```
1       8       16
$       ENDEDIT Null
```

Rule

The $ ENDEDIT directive card must be the last $ card on the *C file. When encountered, the *C file is considered complete.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

$ INCLUDE

Function

The $ INCLUDE directive is used to insert new program elements (source programs, object programs, and/or associated control cards) onto the new source and/or object libraries. The $ INCLUDE directive causes insertion onto the designated library all following program elements up to the next directive control card.

Standard Form

```
1       8       16
$       INCLUDE Option,Option, Additional Options
                               (See Chapter 10)
```

	Options	Meaning
First Option	SOURCE	The source program(s) which follows this directive (up to the next directive) is inserted onto the New Source Library.
	NOSOURCE	The source library is not affected.
Second Option	OBJECT	The object program(s) and/or associated control card(s) which follows this directive (up to the next directive) is inserted into the New Object Library.
	NOBJECT	The object library is not affected.

Examples

```
1       8       16
$       INCLUDE SOURCE,NOBJECT
```

Insert the following source program(s) (up to the next directive) onto the New Source Library. The object library is not affected.

```
1       8       16
$       INCLUDE NOSOURCE,OBJECT
```

Insert the following object program(s) and/or control cards (up to the next directive) onto the New Object Library. The source library is not affected.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

```
1       8       16
$       INCLUDE SOURCE,OBJECT
```

The following actions can occur:

1. If the program elements following the directive are source programs, object program elements (resulting from assembly or compilation) are directed onto the New Object Library. The source elements will be directed onto the New Source Library.

2. If the program elements following the directive are object programs and/or control cards, they will be directed onto the New Object Library. The source library is not affected.

3. If the program elements following the directive are a combination of source programs and object programs and/or control cards, the Source Library Editor will separate those program elements; source programs will be directed onto the New Source Library and the resultant object programs will be directed onto the New Object Library. The object program elements which are part of the input combination will also be directed onto the New Object Library. All elements will be directed onto their respective library in order of their occurrences.

Rules

1. Insertion of program elements onto the new library begins at the point where the new library is positioned at the time the $ INCLUDE directive is encountered.

2. If the options NOSOURCE,OBJECT are specified, all elements following the directive (up to the next directive) must be object programs and/or control cards.

3. The types of program elements which may follow the $ INCLUDE directive are: object programs and/or control cards, and source program elements. The allowable types of source program elements are referred at type 1, type 2, and type 3 modification types, as discussed under the $ MODIFY directive below.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

4. The $ INCLUDE directive is not required for the Initialize Function, unless Additional Options are desired, (see "Additional Options", Chapter 10).

$ LIST

Function

The $ LIST directive is a special Source Library Editor directive used to retrieve a source program from the source library and produce a listing of the source content. The generation of the listing can occur in one of two ways:

1. If the compile mode is ON, the retrieved source program is processed via its respective language processor.

2. If the compile mode is OFF, the retrieved source program is processed by the Source Library Editor and a special listing is written. (See Appendix E for sample printout.)

Standard Form

1	8	16
$	LIST	Name, Additional Options (See Chapter 10)

Option	Meaning
Name	Retrieve and process (in the ON or OFF mode) a source on the Old Source Library identified by Name.

Note: All source programs passed over while the library is searched for program Name are automatically written to the New Source Library.

Example

1	8	16
$	LIST	ALPHA

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

A source program on the Old Source Library which has the name ALPHA punched into columns 73-80 of the $ language card image is located and processed. If the compile mode is ON, the source program is assembled or compiled. If the compile mode is OFF, the Source Library Editor produces a special listing of the source content. In either case, all programs passed over in search of (and including) ALPHA are written to the New Source Library.

The $ LIST directive can also be used to specify action for the entire contents of the source library. This can be accomplished by leaving the first option field blank.

1	8	16
$	LIST	Null

This directive form causes all remaining source programs on the source library to be processed as per $ LIST conventions. Again, all source programs are written to the New Source Library.

Rules

1. The search for the source program to be processed via $ LIST conventions begins wherever the input file is positioned at the time $ LIST is encountered.

2. Only one $ LIST with a blank option field for Name is permitted per activity. If $ LIST is used in this form, it must be directly followed by the $ ENDEDIT control card.

$ MODIFY

Function

The $ MODIFY is a special directive used to replace or alter an existing source program on the source library, or an existing object program of control card on the object library. The $ MODIFY is a combination of $ COPY, $ DELETE, and $ INCLUDE directives in that it is used to copy up to (but not including) the named program element to be modified and written to the new library.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

Standard Form

```
1       8       16
$       MODIFY  Option,Option,Name, Additional Options
                                    (See Chapter 10)
```

	Options	Meaning
First Option	SOURCE	The Old Source Library is copied into the New Source Library up to but not including the specified program, and preparations are made to modify the named program.
	NOSOURCE	The source library is not affected.
Second Option	OBJECT	The Old Object Library is copied into the New Object Library up to but not including the specified program, and preparations are made to replace the named program.
	NOBJECT	The object library is not affected.
Third Option	Name	Program identifier in columns 73-80 of the $ language card on the Old Source Library and the $ OBJECT card on the Old Object Library.

Examples

```
1       8       16
$       MODIFY  SOURCE,NOBJECT,ALPHA
```

Copy the Old Source Library onto the New Source Library up to but not including a source program which has the name ALPHA punched into columns 73-80 of the $ language card image. Preparations are made to modify ALPHA as per a specific modification type, as described below.

```
1       8       16
$       MODIFY  NOSOURCE,OBJECT,ALPHA
```

Copy the Old Object Library onto the New Object Library up to but not including an object program which has the name ALPHA punched into columns 73-80 of the $ OBJECT card image. Preparations are made to replace the object program with the object program following the $ MODIFY directive.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

```
1       8       16
$       MODIFY  SOURCE,OBJECT,ALPHA
```

Both the Old Source Library and Old Object Library are copied onto the New Source Library and New Object Library up to but not including the program ALPHA. The source program is modified as per modification type, and the resultant object program replaces the object program on the object library.

The second subfield (OBJECT/NOBJECT option) need not be specified unless the option is contradictory to the OBJECT/NOBJECT option specified on the $ FILEDIT control card.

Examples

```
1       8       16
$       FILEDIT  SOURCE,NOBJECT
```

```
1       8       16
$       MODIFY  SOURCE,,Name
```

A null field implies the option NOBJECT.

```
1       8       16
$       FILEDIT  SOURCE,OBJECT
```

```
1       8       16
$       MODIFY  SOURCE,,Name
```

A null field implies the option OBJECT.

```
1       8       16
$       FILEDIT  SOURCE,OBJECT
```

```
1       8       16
$       MODIFY  SOURCE,NOBJECT,Name
```

The object library is not to be affected.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

However,

1	8	16
$	MODIFY	,,Name

Equivalent to NOSOURCE,OBJECT. The $ FILEDIT card options are not considered for this form.

Variation of Format

The same directive is used to specify action which can affect a $ control card on the object library. The $ control card processing requires a slightly different format.

1	8	16
$	MODIFY	Option,Name

	Options	Meaning
First Option	NOSOURCE or Null	The source library is not affected.
Second Option	Name	The Old Object Library is copied onto the New Object Library up to but not including the control card type as punched into columns 8-13 of the control card image on the object library. Preparations are made to replace the named control card.

Example

1	8	16
$	MODIFY	NOSOURCE,LINK

or

1	8	16
$	MODIFY	,LINK

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

Copy the Old Object Library onto the New Object Library up to but not including a $ control card which has the name LINK punched into columns 8-13 of the control card image. Preparations are made to replace the $ control card with the $ control card which follows the $ MODIFY directive.

Modification Types

Modification types are discussed for the purpose of classifying the types of input structures allowable following a $ MODIFY directive.

Type 1 Modification. Type 1 modification is indicated by the presence of an equal sign (=) on the $ language control card. A unique file code following the equal sign indicates to the Source Library Editor that the source program associated with this $ language card resides on a completely separate, external file. This external file must be available to the editor via a GECOS $ file card ($ TAPE, $ DISC, etc.). The contents of this external file are processed as input to the Source Library Editor. When an EOF is encountered on the file, the editor returns to examine the next directive on the original input file.

Example

```
1       8          16                          73-80
$       MODIFY     SOURCE,,ALPHA
$       GMAP       =XX                         ALPHA
$       (Next Directive)
```

The source program ALPHA on the source library is replaced by the source program on file XX. The object program (if the OBJECT option is used) on the object library is replaced by the resultant object program from the GMAP assembly. If an equal sign is not present on the $ language card, other modification types may occur.

Type 2 Modification. Type 2 modification occurs when a $ language card and source program follow the $ MODIFY directive. This source program is a replacement for the named program on the old library.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

Example

```
1       8          16                              73-80
$       MODIFY     SOURCE,,ALPHA
$       GMAP                                       ALPHA
          .
          . Symbolic or
          . COMDK
          .
$       (Next Directive)
```

The source program ALPHA on the source library will be replaced by the source program following the $ MODIFY directive. The object program (if the OBJECT option is used) on the object library is replaced by the resultant object program from the GMAP assembly.

Type 3 Modification. Type 3 modification is essentially the same as Type 2 modification except that alters accompany the replacement deck.

Example

```
1       8          16                              73-80
$       MODIFY     SOURCE,,ALPHA
$       GMAP                                       ALPHA
          .
          . Symbolic or
          . COMDK
          .
$       UPDATE
$       ALTER      m1,n1
          .
          .
$       ALTER      m2,n2
          .
          .
$       ALTER      m3,n3
          .
          .
$       (Next Directive)
```

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

Type 4 Modification. Type 4 modification occurs when the $ language card following the $ MODIFY is followed by $ UPDATE rather than a source program. These alters are applied to the existing named source program on the Old Source Library and the replacement is effectively an altered version of the original program.

Example

```
1       8         16                        73-80
$       MODIFY    SOURCE,,ALPHA
$       GMAP                                ALPHA
$       UPDATE
$       ALTER     m1,n1
          .
          .
$       ALTER     m2,n2
          .
          .
$       ALTER     m3,n3
          .
          .
$       (Next Directive)
```

The alters are applied to the source program called ALPHA on the Old Source Library to produce a replacement for that program. The object program (if the OBJECT option is used) on the Old Object Library is replaced by the object program resulting from the GMAP assembly.

Type 5 Modification. Type 5 modification occurs when the special editor directives $ SETSQ and $ SQ follow the $ language card. These special directives are described under their respective headings.

Object Level Modification. Although not classified by modification type, object level inputs must conform to a specific structure. Two types of modification can occur for object level inputs.

Example

```
1       8         16                        73-80
$       MODIFY    ,,ALPHA
$       OBJECT                              ALPHA
          .
          .  Replacement
          .  Object Deck
          .
$       DKEND
$       (Next Directive)
```

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

The object program ALPHA on the object library is replaced by the object program following the $ MODIFY directive.

```
1       8         16
$       MODIFY     ,LINK
$       LINK       LNKB,LNKA
$       (Next Directive)
```

The $ LINK control on the object library is replaced by the $ control card following the $ MODIFY.

Rules

1. Copying from the old library to the new library while searching for the named program element on the $ MODIFY directive begins wherever the input file is positioned at the time the $ MODIFY is encountered.

2. The SOURCE/NOSOURCE option must be used in the first option subfield.

3. The control card type/OBJECT option must be used in the second option subfield.

4. Control card type applies only to the object library since the control card type referred to is not present on the source library.

5. Only one modification type is permitted per $ MODIFY directive. (This also applies to object level modification.)

6. When the Type 4 modification is used, it must be remembered that, although the language processor options are taken from the $ language card followin the $ MODIFY directive, the language processor card on the Old Source Library determines which language processor will be used.

7. Alters cannot be applied to Type 1 modification inputs.

8. Alters cannot be applied to Type 3 or Type 4 in the "compile off" mode.

9. It is invalid to modify a $ ENDLD control card.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

$ PATCH

Function

The $ PATCH is a special Object Library Editor directive which instructs the Object Library Editor to copy the Old Object Library onto the New Object Library up to the named object program. The named object program is then copied onto the New Object Library from the $ OBJECT control card through the last binary card of the deck, the octal correction card(s) following the $ PATCH directive are written onto the New Object Library, and the $ DKEND is copied onto the New Object Library.

Standard Form

1	8	16
$	PATCH	Name, Additional Options (See Chapter 10)

Option	Meaning
Name	Program identifier as punched into columns 73-80 of the $ OBJECT card on the Old Object Library.

Example

1	8	16
$	PATCH	ALPHA
...	OCTAL	...
...	OCTAL	...
...	OCTAL	...
$	(Next Directive)	

The Old Object Library will be copied onto the New Object Library up to the object program which has ALPHA punched into columns 73-80 of the $ OBJECT card image; ALPHA is then copied into the New Object Library up to but not including the $ DKEND card. The octal correction cards following the $ PATCH directive are copied onto the New Object Library. When the next directive on the original input is encountered, the $ DKEND card image is written to the New Object Library.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

Rules

1. The $ PATCH directive applies only to object libraries.

2. Octal correction cards following the $ PATCH directive are inserted between the last binary card and the $ DKEND control card. If octal correction cards already exist in the object deck, the new ones are inserted between the last octal correction card and the $ DKEND card.

3. Copying from the Old Object Library onto the New Object Library begins wherever the input file is positioned at the time the $ PATCH is encountered.

$ PUNCH

Function

The $ PUNCH directive is used to retrieve a named source program from a source library or an object program from an object library. The retrieved program is then written to the special punch file, *7. This special punch file can then be used to obtain a hard copy of the retrieved program.

Standard Form

1	8	16
$	PUNCH	Name, Additional Options (See Chapter 10)

Option	Meaning
Name	Program identifier as punched into columns 73-80 of the $ language card on the source library.

This above form is used primarily for source programs on the source library. However, another form of the $ PUNCH directive exists:

1	8	16
$	PUNCH	Option,Option,Name, Additional Options (See Chapter 10)

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

	Option	Meaning
First Option	NOSOURCE	The source library is not affected.
	SOURCE	Retrieve and punch (write to *7) a source program residing on the Old Source Library identified by Name.
Second Option	OBJECT	Retrieve and punch (write to *7) an object program residing on the Old Object Library identified by Name.
	NOBJECT	The object library is not affected.
Third Option	Name	Program identifier as in columns 73-80 of the $ language card on the source library or the $ OBJECT card on the object library.

Examples

```
1        8        16
$        PUNCH    ALPHA
```

or

```
1        8        16
$        PUNCH    SOURCE,NOBJECT,ALPHA
```

Retrieve a source program residing on the source library which has the name ALPHA punched in columns 73-80 of the $ language card image. Write this program (including the $ language card) to the *7 punch file. The object library is not affected.

```
1        8        16
$        PUNCH    NOSOURCE,OBJECT,ALPHA
```

Retrieve an object program from the object library that has the name ALPHA punched into columns 73-80 of the $ OBJECT card. Write this program to the *7 punch file. The source library is not affected.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

```
1       8       16
$       PUNCH   SOURCE,OBJECT,ALPHA
```

Retrieve a source program from the source library that has the name ALPHA punched into columns 73-80 of the $ language card; retrieve an object program from the object library which has the name ALPHA punched into columns 73-80 of the $ OBJECT card. Both programs are written to the *7 punch file.

The second subfield (OBJECT/NOBJECT option) need not be specified unless the option is contradictory to the OBJECT/NOBJECT option specified on the $ FILEDIT card.

Examples

```
1       8       16
$       FILEDIT SOURCE,NOBJECT
```

```
1       8       16
$       PUNCH   SOURCE,,Name
```

A null field implies the NOBJECT option.

```
1       8       16
$       FILEDIT SOURCE,OBJECT
```

```
1       8       16
$       PUNCH   SOURCE,,Name
```

A null field implies the OBJECT option.

```
1       8       16
$       FILEDIT SOURCE,OBJECT
```

```
1       8       16
$       PUNCH   SOURCE,NOBJECT,Name
```

It is necessary to specify the NOBJECT option to imply that the object library is not affected.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

1	8	16
$	PUNCH	,,Name

Only the object library will be affected. The $ FILEDIT options are not considered for this form.

Rules

1. As the Old Source Library or Old Object Library is searched for the named program to be punched (written to *7), all contents (including the named program) are written to the New Source Library or New Object Library. Copying from the old library to the new library begins wherever the input file is positioned at the time $ PUNCH is encountered.

2. When $ PUNCH is used, a file for *7 ($ SYSOUT..*7, or $ TAPE...*7) must be supplied in the main part of the editor deck setup. For small volumes of punched output, *7 may be assigned to SYSOUT. For large volumes of punched output, *7 should be assigned to tape; output can then be punched by using BMC.

3. The *7 file cannot be assigned to a physical punch device.

4. Each source program written to *7 is preceded by its respective $ language card.

5. Each source program written to *7 includes its $ OBJECT and $ DKEND cards. Octal correction cards, if present within an object deck, are also written to *7.

6. When both source programs and object programs are written to *7, all source programs appear first, followed by all object programs.

7. A single octal 17 EOF terminates the *7 file.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

$ SEQ

Function

The $ SEQ directive is used with the $ MODIFY directive to add or delete source cards from a source program residing on the Old Source Library. The $ SEQ function is identical to the $ ALTER function except that sequence identifiers are punched into columns 73-80 of each source statement card within the source program.

Since the $ SEQ card is a subdirective of the $ MODIFY directive, it is discussed as a control type rather than directive function.

Standard Form

1	8	16
$	SEQ	m,n

where m = beginning sequence identifier
n = ending sequence identifier

When it is desired simply to insert source statement cards into a source program, the n subfield is not used. In this case, the cards following the $ SEQ card, up to but not including the next $ SEQ card (or next directive) use inserted immediately prior to the card corresponding to sequence identifier m.

Example

1	8	16	73-80
$	MODIFY	SOURCE,,ALPHA	
$	GMAP		ALPHA
$	SEQ	ALPHA010	
	. Source cards		
	. to be		
	. inserted		
$	(Next Directive or $ SEQ)		

The source cards following the $ SEQ card are inserted into the existing source program ALPHA immediately prior to the source statement card which has the identifier sequence ALPHA010 punched into columns 73-80 of the source card image.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

10. ADDITIONAL OPTIONS

Additional options are special subdirective options which enable to secondary function to occur while a primary function is being executed. An example would be the ability to list and punch the same source program within a single edit run. (This previously required two edit runs.)

The discussion of additional options was purposely withheld until this chapter in hopes that the user may become completely familiar with the editor directives and the standard options (as described in Chapter 9) before attempting to take advantage of the additional options feature.

Additional options which are available with certain editor directives described in Chapter 9 are as follows:

BCD - This option can be used to cause a source program (as it is being processed) to be written to the *7 punch file in the BCD (Hollerith) format rather than the COMDK (compressed deck) format.

COMDK - This option can be used to cause a source program (as it is being processed) to be written to the *7 punch file in the COMDK format.

DECK - This option can be used to cause an object program (as it is being processed) to be written to the *7 punch file.

IBMC - This option can be used to cause the following action to occur to a source program (as it is being processed):

1) The source program is written to the *7 punch file.

2) The source program is written in the BCD format.

3) All characters are transliterated (decoded) into the IBMC character set.

IBMEL - Same as IBMC except all characters are transliterated into the IBMEL character set.

IBMF - Same as IBMC and IBMEL except all characters are transliterated into the IBMF character set.

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

ON - The compile mode indicator is set to the ON mode.

OFF - The compile mode indicator is set to the OFF mode.

The following discussion pertains only to those editor directives for which additional options are applicable. Rules that apply directly to the specific directive are discussed with the directive. General rules applying to all additional options appear at the end of this chapter.

$ INCLUDE

All additional options are available with this directive. The following rule applies specifically to additional option usage for $ INCLUDE:

When additional options are to be used, the first and second options subfields (see $ INCLUDE discussion, Chapter 9) must be used.

```
Allowable:      $ INCLUDE  SOURCE,,COMDK
Not Allowable:  $ INCLUDE  SOURCE,COMDK
```

$ LIST

All additional options, with the exception of DECK, are available with this directive. The following rules apply specifically to additional option usage for $ LIST:

1. If the compile mode is OFF, the named program on the Old Source Library is decompressed and a special BCD listing of the source content is written to P*. (See Appendix E for a sample printout of this report.)

2. If the compile mode is ON, the named program on the Old Source Library is processed by its respective language processor and an assembly or compilation listing is written to P*.

3. If the entire contents of the source library are to be listed (in the ON or OFF mode) and any additional options are to apply, the first option subfield (see $ LIST discussion, Chapter 9) must be null.

```
Allowable:      $ LIST  ,OFF,COMDK
Not Allowable:  $ LIST   OFF,COMDK
```

4.52 SYSTEMS INFORMATION BULLETIN #57 (Cont.)

$ MODIFY

All additional options are available with this directive. Rules pertaining to additional option usage with $ MODIFY are discussed under general rules at the end of this chapter.

$ PATCH

DECK is the only additional option available with the $ PATCH directive.

$ PUNCH

The ON/OFF option is ignored. All other additional options can be used with $ PUNCH. Rules pertaining to additional option usage for the $ PUNCH directive are discussed below.

General Rules for Additional Options

1. When the compile mode if OFF, additional punch options (COMDK or BCD) are processed as the source program is being copied to the New Source Library.

2. When the compile mode is ON, additional punch options (COMKD or BCD) are porcessed upon return of control from the respective language processor. Additional punch options in the compile ON mode require the special alternate COMDK file (*K).

3. Additional options may appear in any order within the additional options subfield. The first blank character encountered terminates the additional options scan.

4. Non-recognizable options are ignored.

5. Any additional option specified which is contradictory to the main directive is ignored.

   Example: $ MODIFY ,,ALPHA,ON,COMDK,DECK

   In the above example, the options ON and COMDK are ignored because the $ MODIFY directive specifies action for the object library only.

4.53 SYSTEMS INFORMATION BULLETIN #58 8/11/71

SUBJECT: Known Error in the FILEDITOR in SDL 3.3

Character transliteration is not performed correctly when the $ INCODE control card is placed on the *C file. An alternative method using a BMC activity to do the transliteration is listed below.

```
$    IDENT
$    CONVER
$    READ     IN,A2R
$    TAPE     OT,X4S
$    INPUT    IBMF,MBCD
$    FILEDIT  SOURCE,NOBJECT,=FC
$    LIMITS   999,44000,0,99999
$    SYSOUT   FC
$    TAPE     K*,X2R
$    TAPE     M*,X3D,,,,SOURCE-INPUT
$    MASS     G*,A3R,100L
$    MASS     *1,A1R,100L
$    TAPE     *C,X4D
$    ENDJOB
***EOF
```

INPUT DECK FOR FIRST ACTIVITY:

```
$    MODIFY   SOURCE,NOBJECT,NAME
$    GMAP                              NAME
$    UPDATE   LIST
$    ALTER
. . . . . .
$    ENDEDIT
***EOF
```

4.54 SYSTEMS INFORMATION BULLETIN #59 8/19/71

SUBJECT: Real Time File Code Error in SDL-3.3.2

File codes R2, R7, S0 and S1 should not be used in SDL-3.3.2 by real time programs. This will be corrected in SDL-3.3.3.

A stupid mistake by IN-DAT-41 resulted in the real time file code table.

4.55 SYSTEMS INFORMATION BULLETIN #60 8/31/71

SUBJECT: MME RTBORT

A new real-time Master Mode Entry, MME RTBORT, has been added to GSEC. This MME allows a real-time programer to request a slave abort. A post-mortem dump of the program is printed if the option DUMP is specified on the $ EXECUTE control card. The abort reason code will be the last two BCD characters in the Q-Register. If the address bits (0-17) of word 23 (27 octal) of the Slave Program Prefix are within the slave program limits, return is made to the user for wrap-up after leaving real-time mode. All I/O activity should be terminated prior to executing this MME or the I/O may be lost.

4.56 SYSTEMS INFORMATION BULLETIN #61 9/1/71

SUBJECT: COBOL Print Line Length

The standard COBOL print line length has been extended, in SDL 3.3, from a maximum of 132 character positions (in multiples of 6) to a maximum of 136 positions. Note that 133, 134, or 135 position lines should not be used as zero fill to position 136 will occur.

Two problems have been noted by this. The first results from any program iuing 133 to 135 position lines. Previously truncation at position 132 occurred. Now the filled zeros are printed.

The second problem occurred when the Phoenix patches erroneously created a useless 24th word in the print buffer when 23 words were required to fill the line. Programs that used a 132 character line and picked up the slew code from the last word of the record were getting an overprint in some cases. This problem is corrected in SDL 3.3.4 by a software change eliminating the useless 24th word.

4.57 SYSTEMS INFORMATION BULLETIN #62 11/4/71

SUBJECT: LIMITS Control Card

During the past year, several changes have been made to the LIMITS Control Card, and the limits set by the system in the absence of a LIMITS Card have been lowered. These changes are described in the Attachment.

4.57 SYSTEMS INFORMATION BULLETIN #62 (Cont.)

$ LIMITS

1	8	16
$	LIMITS	Time,Storage 1,Storage 2,Print Lines, I/O Time

Function:

The $ LIMITS control card is used to modify standard activity limits. If omitted, allocation will be made as indicated in Rule 4.

Operand Field:

Five entries are contained in the operand field.

Time, the first entry, specifies the manimum processor run time for the activity, expressed in hundredths of an hour. Beyond this limit, the job is aborted. If an activity calls upon a system program, (i.e., an object program calling SORT), this processing time must include all the time estimated for the activity. The maximum value that can be punched in this field is 999 (9.99 hours); the minimum, 0.01.

Storage 1 denotes the maximum core storage requested for running the program. The units of storage are decimal digits representing the number of words desired. (Actual memory allocation is made in multiples of 1024 words). The smallest request that will be allocated is 1024 words (1K). If a value is followed by the character "K", this value is multiplied by 1024 before its use.

Storage 2 indicates the amount of core storage requested by a slave program that may be shared with GELOAD when the program is loaded. An explanation of the types of memory usage which may be shared with GELOAD is given below.

4.57 SYSTEMS INFORMATION BULLETIN #62 (Cont.)

An execution activity may be broken into two phases; loading and execution. During loading, GELOAD places information into preassigned areas of the user's memory. There may be other areas, such as LABELED COMMON regions, buffers, and BLANK COMMON into which no information is being loaded at load time but which will be used by the program during execution. By proper use of control cards ($ USE, $ EQUATE, $ OPTION), all of these regions may be assigned to the low end addresses of the user's memory such that the area which they represent may be shared with GELOAD during loading. When execution begins, GELOAD (.SETU. routine will have cleared this area for use by the program.

A minus sign may precede the value specified in Storage 2. If so, value specified is added to the size of GELOAD to allow extra space for load tables.

Print Lines specify the maximum number of lines to be written on SYSOUT during program execution for last printing.

I/O Time may be used to limit the accumulated I/O time within any activity. This time is given in hundredths of hours. When this field is not given, it implies unlimited I/O time. Presence of the field will cause the total channel busy time, regardless of the device type, to be limited to the specified amount.

Example:

1	8	16
$	LIMITS	10,10000,0,1000,05

Maximum Run Time 0.10 hr.
Maximum Storage 10,000 words
No memory can be shared
Maximum of 1,000 lines written on SYSOUT
Maximum channel busy time 0.05 hr.

4.57 SYSTEMS INFORMATION BULLETIN #62 (Cont.)

Rules

1. In a Compile and Go or Assemble and Go operation, the $ LIMITS card following the $ EXECUTE card defines the limits for the anticipated execution of the user's program and not the compilation or assembly process.

2. The $ LIMITS card is not required for system programs such as FORTRAN or COBOL, since standard limits are predefined. If these limits are to be modified, however, the $ LIMITS card should follow immediately behind the system call card ($ FORTRAN, $ GMAP, $ COBOL, etc.).

3. When the $ LIMITS control card is omitted, the following processor run time, memory and SYSOUT lines limits are assumed. The I/O time is not limited.

Activity	Hr.	Memory+SSA	Lines
ALGOL	0.08	28k	10,000
CONVER	0.08	6k	1,000
COBOL	0.15	32k	20,000
EXECUTE	0.05	16k	5,000
EXTEDIT/FILEDIT	0.04	32k	10,000
FORTRAN	0.05	25k	12,000
FILSYS	0.03	32k	1,000
GMAP	0.04	24k	10,000
IDS	0.15	32k	20,000
JOVIAL	0.08	28k	10,000
PROGRAM	0.05	16k	5,000
PRODUCT	0.50	20k	30,000
SYSEDIT	0.05	32k	10,000
UTILITY	0.03	8k	10,000
1401SIM	0.08	12k	5,000
225SIM	0.08	24k	5,000
44SIM/94SIM	0.08	48k	5,000

Compilers, assemblers and executions whose limit is more than 2500 lines of SYSOUT require 2k SSA; less than 2500 lines, 1k SSA.

5. Storage 2 may contain any number. However, if this number is less than 9k (size of the Loader), the Allocator allocates the extra core needed to load the program. If the number is greater then 9k, the Allocator reduces this number to 9k. If a minus sign precedes the overlay value, the value is added to 9k to allow extra space for load tables.

4.57 SYSTEMS INFORMATION BULLETIN #62 (Cont.)

6. Fields can be explicitly nulled:

$ LIMITS 5,,,5000

which will result in the standard limits for the activity in question where the fields are null.

7. When the lines to be output by SYSOUT are limited due to the above specified circumstances, the lines counted include only those supplied by the slave program. Banner lines, GEPR error messages and the accounting report, are not counted. GEBORT core dumps and GESNAP print lines are counted as described in CPB-1518.

8. A $ LIMIT card placed in the job deck ahead of the first activity definition card ($ FORTRAN, $ EXECUTE, etc.) is interpreted as a job limit definition. The processor run time, I/O time and SYSOUT line limit fields will be used to limit the job as a whole. The actual resources used in each activity are subtracted from the job total. The job will be terminated when the resource left becomes zero.

4.58 SYSTEMS INFORMATION BULLETIN #63 12/16/71

SUBJECT: Use of DISC & DRUM in RT

Random disc and/or drum files may be used in real time. The 5-word calling sequence must be used for these random files.

```
            MME             RTINOS
            SDRA
            ZERO            FC, DCW1
            RDR or WDR
            ZERO            FC, DCW2
            ZERO            ST, CC
             .
             .
             .
DCW1        IOTD            SKADD, 1
SKADD       OCT             0           Block number

FC          BCI             1, 0000FC
ST          BSS             2           Same use as in GECOS
DCW2        IOTD            DATA, n     calling sequence
CC          NULL
```

4.58 SYSTEMS INFORMATION BULLETIN #63 (Cont.)

NOTES: (1) These devices (disc and drum) are shared devices; thus, RT doesn't have exclusive use of the channel. RT will, however, become next in line to start I/O, if I/O is in progress.

(2) One block = 64 words.

(3) Block 0 is the first block.

(4) Attempt to start access outside file space will result in 17 abort.

(5) If I/O starts in the file but goes beyond file limits, error status will be returned.

4.59 SYSTEMS INFORMATION BULLETIN #64 12/16/71

SUBJECT: Loading H* Links in Real Time

RTRSTR is a real time subroutine which performs a function similar to GERSTR; namely, to load a line (overlay) from an H* file created by GELOAD. H* may be on drum, disc, or tape.

Calling Sequence:

Q-reg 24-35 = file code

```
L       TSXI    RTRSTR
L+1     BCI     1, name
L+2     ZERO    A
L+3     ZERO    B
L+4     NULL
```

name = name of link to be loaded

A = load location. If A = 0, use the same location used when the file was created.

B = courtesy call address, i.e., the location to which control will be given if loading is successful. If B = 0, control will be given to the link at its entry point. (The user must have $ ENTRY card to specify an entry point).

4.59 SYSTEMS INFORMATION BULLETIN #64 (Cont.)

(1) On Return With No Errors:

AU = starting load address
AL = number of words
QU = entry point

(2) On Error Return

A = 0 and Q = error code:

Q = 1 if tape format error
Q = 2 if tape error
Q = 3 if invalid file code
Q = 4 if tape error
Q = 5 if checksum error
Q = 6 if name not in catelog

NOTES:

1. If no $ ENTRY card is used, the entry point will be 0.
2. The format of H* files on tape is not the same as disc and drum.
3. The disc and drum files must be random.
4. The routine is courtesy call driven.
5. Control will be returned to location L+4. Here the user should check the A register for errors and relinquish until I/O is complete if no error.
6. The user should also check the A register for errors when the courtesy call is initiated.

C.4